



AFRL-RI-RS-TR-2014-108

## **SYNTHESIS OF ROAD NETWORKS BY DATA CONFLATION**

---

RESEARCH FOUNDATION OF SUNY

*APRIL 2014*

FINAL TECHNICAL REPORT

***APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED***

STINFO COPY

**AIR FORCE RESEARCH LABORATORY  
INFORMATION DIRECTORATE**

## **NOTICE AND SIGNATURE PAGE**

Using Government drawings, specifications, or other data included in this document for any purpose other than Government procurement does not in any way obligate the U.S. Government. The fact that the Government formulated or supplied the drawings, specifications, or other data does not license the holder or any other person or corporation; or convey any rights or permission to manufacture, use, or sell any patented invention that may relate to them.

This report was cleared for public release by the 88<sup>th</sup> ABW, Wright-Patterson AFB Public Affairs Office and is available to the general public, including foreign nationals. Copies may be obtained from the Defense Technical Information Center (DTIC) (<http://www.dtic.mil>).

AFRL-RI-RS-TR-2014-108 HAS BEEN REVIEWED AND IS APPROVED FOR PUBLICATION  
IN ACCORDANCE WITH ASSIGNED DISTRIBUTION STATEMENT.

FOR THE DIRECTOR:

**/ S /**

ADNAN BUBALO  
Work Unit Manager

**/ S /**

MICHAEL J. WESSING  
Deputy Chief, Information Intelligence  
Systems and Analysis Division  
Information Directorate

This report is published in the interest of scientific and technical information exchange, and its publication does not constitute the Government's approval or disapproval of its ideas or findings.

REPORT DOCUMENTATION PAGE				Form Approved OMB No. 0704-0188	
<p>The public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Department of Defense, Washington Headquarters Services, Directorate for Information Operations and Reports (0704-0188), 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302. Respondents should be aware that notwithstanding any other provision of law, no person shall be subject to any penalty for failing to comply with a collection of information if it does not display a currently valid OMB control number.</p> <p><b>PLEASE DO NOT RETURN YOUR FORM TO THE ABOVE ADDRESS.</b></p>					
1. REPORT DATE (DD-MM-YYYY) APRIL 2014		2. REPORT TYPE FINAL TECHNICAL REPORT		3. DATES COVERED (From - To) DEC 2010 – DEC 2013	
4. TITLE AND SUBTITLE  SYNTHESIS OF ROAD NETWORKS BY DATA CONFLATION				5a. CONTRACT NUMBER FA8750-11-2-0082	
				5b. GRANT NUMBER N/A	
				5c. PROGRAM ELEMENT NUMBER 62788F	
6. AUTHOR(S)  Tarunraj Singh				5d. PROJECT NUMBER E2DT	
				5e. TASK NUMBER UB	
				5f. WORK UNIT NUMBER RE	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Research Foundation of SUNY Sponsored Projects Services 402 Crofts Hall Buffalo, NY 14260-0001				8. PERFORMING ORGANIZATION REPORT NUMBER	
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)  Air Force Research Laboratory/RIEA 525 Brooks Road Rome NY 13441-4505				10. SPONSOR/MONITOR'S ACRONYM(S) AFRL/RI	
				11. SPONSOR/MONITOR'S REPORT NUMBER AFRL-RI-RS-TR-2014-108	
12. DISTRIBUTION AVAILABILITY STATEMENT  Approved for Public Release; Distribution Unlimited. PA# 88ABW-2014-1857 Date Cleared: 22 APR 2014					
13. SUPPLEMENTARY NOTES					
14. ABSTRACT For application including post disaster scenarios or unmapped areas of conflict, there is a need for rapid real-time synthesis of road networks. GMTI data in conjunction with the assumption that the vehicles move on roads, permit the use of target kinematics to estimate the underlying road network that the vehicles are constrained to. The approach employed for synthesizing the received data into a complete estimate of the road network is through the use of the Hough Transform, to identify line segments which collectively represent the road network. The Total Least Squares is used to characterize the uncertainty associated with this representation as well as provide a more accurate estimate. The uncertainty in each of the estimated parameters can then be approximated by the Cramer Rao lower bounds. Finally the identified segments are merged and connected to provide a more complete representation of the road network. The approach used here is iterative, for example, when new data within the area of interest is received, a better estimate of the road segment can be obtained and the overall road network is updated and allowed to grow in all direction.					
15. SUBJECT TERMS Road Network Estimation; Target Tracking, Hough Transform, Total Least Squares, GMTI Tracks					
16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT  UU	18. NUMBER OF PAGES  121	19a. NAME OF RESPONSIBLE PERSON ADNAN BUBALO
a. REPORT U	b. ABSTRACT U	c. THIS PAGE U			19b. TELEPHONE NUMBER (Include area code) N/A

## TABLE OF CONTENTS

List of Figures.....	iv
List of Tables.....	vii
Acknowledgements.....	vii
1.0 SUMMARY.....	1
2.0 INTRODUCTION.....	2
2.1 Related Work.....	2
2.2 Overview.....	15
3.0 METHODS.....	17
3.1 Hough Transform.....	17
3.2 Line Hough Transform.....	17
3.3 Circle Hough Transform.....	21
3.4 Ellipse Hough Transform.....	30
3.5 Maximum Likelihood Estimators.....	33
3.6 Least Squares.....	33
3.7 Total Least Squares.....	34
3.8 LS and TLS Comparison.....	38
3.9 Ellipse Fitting.....	41
3.10 LS Ellipse Example.....	46
3.11 Uncertainty Analysis.....	47
3.12 Straight Line.....	47
3.13 Algebraic Fit Covariance.....	55
3.14 3rd Order Polynomial.....	57
3.15 Ellipse.....	61
4.0 ASSUMPTIONS AND PROCEDURES.....	67
4.1 Overview.....	67
4.2 Load Database.....	69
4.3 Line Extraction.....	70
4.3.1 Slider.....	70
4.3.2 Step Size.....	70
4.3.3 Distance Threshold.....	71
4.3.4 Hough Parameters.....	71
4.3.5 Step.....	72
4.3.6 Recursive.....	75
4.4 Manual Intervention.....	75
4.4.1 Merge.....	76
4.4.2 Ellipse.....	77
4.4.3 Remove.....	80
4.5 Post Processing.....	81
4.5.1 Blend.....	81
4.5.2 Trim/Extend.....	83
4.5.3 Uncertainty Computation.....	84
4.5.4 Extract.....	85
4.6 Stored Structure.....	86
4.6.1 Export Data.....	86

4.7 Additional Features.....	88
4.7.1 Reset.....	89
4.7.2 Import Data.....	89
4.7.3 Image Options.....	89
5.0 RESULTS.....	90
5.1 Synthetic Data.....	90
5.2 Data Set #1 Results.....	92
5.2.1 Graphical Results.....	93
5.2.2 Uncertainty Analysis.....	95
5.3 Data Set #2 Results.....	99
5.3.1 Graphical Results.....	100
5.3.2 Uncertainty Analysis.....	104
6.0 CONCLUSIONS.....	108
References.....	110
List of Acronyms.....	112

## LIST OF FIGURES

1	MHT Algorithm [3] pp. 7 . . . . .	3
2	Linearity Determination [2] pp. 203 . . . . .	4
3	Pixel Templates [5] pp. 91 . . . . .	5
4	Region Locations [6] pp. 1194 . . . . .	7
5	Algorithm Outline [8] pp. 1785 . . . . .	9
6	Footprints [9] pp. 4147 . . . . .	11
7	Toe-Finding Algorithm [9] pp. 4148 . . . . .	12
8	New Track Processing [10] pp. 1170 . . . . .	13
9	Parameter Identification . . . . .	18
10	Collinear Points . . . . .	19
11	Linear Transformation of $\rho$ . . . . .	19
12	Hough Transform . . . . .	20
13	Circle Parameter Identification . . . . .	21
14	Accumulation Matrix . . . . .	22
15	Unequal Non-concentric Circles . . . . .	23
16	Simple CHT Algorithm Results . . . . .	25
17	Tao Peng Example 2 . . . . .	27
18	Results from Tao Peng's CHT Algorithm . . . . .	28
19	Results from David Young's CHT Algorithm . . . . .	29
20	Candidate Ellipse Points [14] pp. 778 . . . . .	30
21	Ellipse Center Location [15] . . . . .	31
22	Least Squares Minimization . . . . .	34
23	Total Least Squares Minimization . . . . .	35
24	Perpendicular Distance Point to Line [20] . . . . .	39
25	Least Squares and Total Least Squares Comparison (zoom) . . . . .	40
26	Least Squares Ellipse Fitting Results . . . . .	46
27	Truth, Measurements, Estimate, TLS Line Fit . . . . .	52
28	Monte Carlo Simulation . . . . .	53
29	Sigma Ellipses . . . . .	54
30	One Sigma Slopes/Intercepts . . . . .	55

31	3rd Order Polynomial Simulation . . . . .	58
32	3rd Order Polynomial Uncertainty . . . . .	59
33	3rd Order Polynomial Uncertainty Convergence . . . . .	60
34	3-Sigma Polynomials . . . . .	61
35	Ellipse Simulation . . . . .	63
36	Ellipse Axis Uncertainty . . . . .	64
37	Ellipse Uncertainty Convergence . . . . .	65
38	3-Sigma Ellipses . . . . .	66
39	Graphical User Interface . . . . .	68
40	Algorithm Overview . . . . .	68
41	Load Database Change . . . . .	70
42	Hough Parameters . . . . .	71
43	10 Tracks Processed . . . . .	75
44	Merge Selection . . . . .	76
45	Merge Results . . . . .	77
46	Ellipse Results . . . . .	78
47	Removal of Ellipse . . . . .	81
48	Polynomial Blend . . . . .	82
49	Trimmed/Extended Features . . . . .	84
50	Image Options . . . . .	89
51	Data Set #1 Measurements . . . . .	91
52	Data Set #2 Measurements . . . . .	92
53	Data Set #1 Phase One . . . . .	93
54	Data Set #1 Phase Two . . . . .	94
55	Data Set #1 Phase Three . . . . .	94
56	Data Set #1 Final Extraction Results . . . . .	95
57	Data Set #1 Line CRLB . . . . .	96
58	Data Set #1 Error Ellipses One Sigma . . . . .	97
59	Data Set #1 Mean Line Uncertainty . . . . .	98
60	Data Set #2 Phase One . . . . .	100
61	Data Set #2 Phase Two . . . . .	101

62	Data Set #2 Phase Three . . . . .	102
63	Data Set #2 Phase Four . . . . .	102
64	Data Set #2 Phase Five . . . . .	103
65	Data Set #2 Phase Six . . . . .	103
66	Data Set #2 Final Extraction Results . . . . .	104
67	Data Set #2 Line CRLB . . . . .	105
68	Data Set #2 Error Ellipses One Sigma . . . . .	106
69	Data Set #2 Mean Line Uncertainty . . . . .	107



## LIST OF TABLES

I	CHT Numerical Comparison . . . . .	32
II	Parameter Values and RMSE . . . . .	47

## Acknowledgements

This material is based on research sponsored by Air Force Research Laboratory under Cooperative Agreement number FA8750-11-2-0082. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation thereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of Air Force Research Laboratory or the U.S. Government.

## 1.0 SUMMARY

Road maps can provide information on not only topographical conditions but also on how materials and people move. Historically, available information pertaining to road maps consisted of static images of preordained routes. Even now, Google maps and hand held Global Positioning Systems (GPS) represent a static view of road networks, requiring either recapturing images or manually updating units. However, in order to have more current, information rich representations of transportation networks or road maps, the use of kinematic information provided by sensors such as Ground Moving Target Indicator (GMTI) data is required. The data these GMTI sensors supply is not only a static representation of a target but also the kinematics. The approach employed for synthesizing the received data into a complete estimate of the road network is through the use of the Hough Transform, to identify line segments which collectively represent the road network. The Total Least Squares is used to characterize the uncertainty associated with this representation as well as provide a more accurate estimate. The uncertainty in each of the estimated parameters (i.e. slope and intercept of a line) can then be approximated by the Cramer Rao lower bounds. Finally the identified segments are merged and connected to provide a more complete representation of the road network. The approach used here is iterative, for example, when new data within the area of interest is received, a better estimate of the road segment can be obtained and the overall road network is updated and allowed to grow in all directions.

## 2.0 INTRODUCTION

The need to develop accurate estimates for road networks is important in both military and civilian applications. Synthetic Aperture Radar (SAR) and Ground Moving Target Indicator (GMTI) data is often processed and analyzed to produce such networks. SAR produces images of varying intensity which can be processed to separate buildings, roads, and terrain. However, SAR is only able to detect prominent existing features [1]. In other words, SAR will only detect a road if there is a distinct outline of such a path. GMTI on the other hand tracks moving targets and relays the latitude and longitude coordinates as well as the velocity in both directions. The disadvantage of GMTI however, is the necessity of a moving target, should the target stop or be obstructed in any way from the sensors, the tracker will lose the target for the duration of the obstruction [2]. Many of the currently available algorithms rely on information from pre-existing road maps [1], [3], [5], [7]], however in many scenarios the availability of this a-priori information is limited and inaccurate. In additional situations there is no existence of road maps, such as in times of conflict in desert regions. Therefore the need for an algorithm which can accurately estimate road networks in a timely manner is of great importance. Furthermore, there is a lack of a quantifiable measure of the accuracy of the extracted road estimates. Several available algorithms use a "completeness" and "correctness" measure which is a comparison of the extracted road network and the actual network [7], [8], [9], [11]], however as previously stated in many situations there is no available truth so these methods are not relevant.

### 2.1 Related Work

Koch, Koller and Ulmke utilize Multiple-Hypothesis Tracking (MHT) to develop the estimates of the target state vector  $\mathbf{x}_k$  which is composed of the ground coordinates,  $\mathbf{r}_k$ , and their component velocities,  $\dot{\mathbf{r}}_k$ , at time  $k$  [2]. In most scenarios the z-component of both the ground coordinates and the velocity are ignored or set to zero. The MHT algorithm is outlined in Figure 1.

MHT is a stochastic logic decision maker based on both current and future observations. Suppose there exist two tracks, and at the time step,  $k$ , three observations are received. Multiple hypotheses are made as to which tracks these observations correspond to, either one track, both tracks, or neither of the tracks are plausible hypotheses. Each of these hypothesis are then propagated into the future, and once additional observations are obtained these hypotheses are evaluated and the

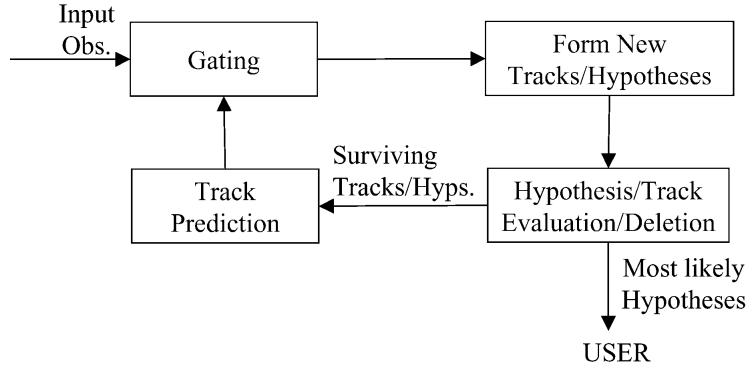


Fig. 1: MHT Algorithm [3] pp. 7

most probable track is accepted. The first box in Figure 1, termed "Gating", forms a region based on the covariance matrices of the predictions and only hypotheses within this region are accepted, this reduces the initial number of hypotheses. For the MHT algorithm the target state is modeled by a linear Markov process which is defined such that the state at time  $t_{k+1}$  is dependent on only the preceding time step,  $t_k$ . The measurement function is also assumed to be linear in terms of the state. Retro-diction allows for the smoothing of the hypotheses, it does this by recalculating the previous estimates with all available measurements. In this algorithm all roads are modeled by linear segments, additional segments are added depending on the linearity of the road. In order to describe the linearity of the road, the directional vector of the velocity,  $\dot{\mathbf{r}}_{l|k}$  and the difference in the ground coordinates,  $\mathbf{r}_{l+1|k} - \mathbf{r}_{l|k}$  are considered. Where  $l$  denotes the node and  $k$  is the time step. Figure 2 depicts a graphical representation of a curved road segment where additional nodes must be incorporated.

The angle of the distance vector,  $\psi_{l|k}$  and the angle made by the tangential velocity vector and the distance vector, denoted by,  $\phi_{l|k}$  are used to develop Equation 1, the decision criteria for including additional nodes for line segments.

$$\frac{(\phi_{l|k} - \psi_{l|k})^2}{\Phi_{l|k}} > \kappa^2 \quad (1)$$

Where  $\kappa$  is the decision parameter and after experimentation is selected to be one and  $\Phi_{l|k}$  denotes the covariance matrix of the velocity.

Baumgartner, Hinz, and Wiedemann developed a semi-automatic method for line extraction from images [4]. The user is required to identify an initial segment which the tracker will

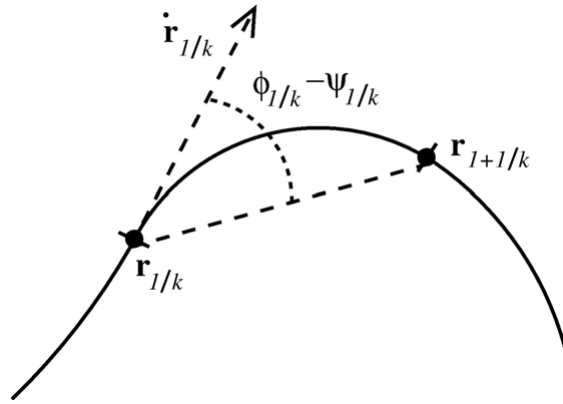


Fig. 2: Linearity Determination [2] pp. 203

use to then grow the entire network and should the automatic tracker encounter an error, the user is required to take action. Two main classes of problems can be encountered during the implementation of the road tracking algorithm. The first case is when the tracker reaches the edge of the image. The second case is when the internal confidence of the tracker is below a desired level. This algorithm works well for distinct road like features within the image. There are five steps performed during the course of the estimation of the road network defined below:

- 1) Generate a reference profile
- 2) Predict the next position of the road axis
- 3) Estimate the position of the next point on the road
- 4) Check for stopping criteria, if none return to (2) otherwise (5)
- 5) Request user interaction

This process is repeated until all desired roads have been identified.

Zhou, Venkateswar, and Chellappa begin the extraction of roads with an edge detector to identify the boundaries of such roads [5]. This produces edge pixels which are then utilized to identify linear features in an image. The linear feature extraction is broken up into three steps. First the gradient direction of the edge pixels must be determined. For simplification purposes, the gradient direction is discretized into eight evenly spaced values between -180 and 180 degrees. To determine whether two edge pixels are collinear the template of the edge pixel is compared against a database of available templates. The templates for four of the eight gradient directions are shown in Figure 3.

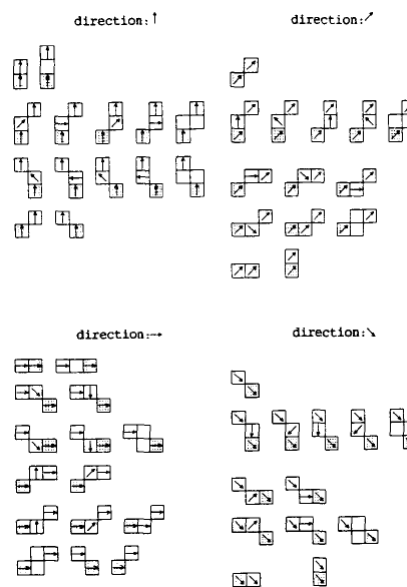


Fig. 3: Pixel Templates [5] pp. 91

Each edge pixel is then separated into the corresponding line structure. The second step is to connect the identified lines. A small neighborhood is examined around each of the endpoints of

a line. Two criteria must be met in order for the lines to be merged. First the direction of the second line must be similar to the first, and second, the magnitude of the second line must lie between half and twice the magnitude of the first line. This step is performed twice, once for a direction delta of 8 degrees and second for a delta of 13 degrees. The third and final step in the extraction is the extension of line segments to form corners. This is easily done by determining if two lines will intersect due to their directional gradients and if so, they are extended until they intersect. This corner extraction method however is only viable for two intersecting lines, should there be more, the corner will fail to be an accurate estimate.

Amo, Martinez, and Torre extract roads from aerial images using a region competition algorithm [6]. This method is semi-automatic in which the user selects points along the road. The amount of points required varies directly with the curvature of the road segment. These user selected points must be located at large curvature changes and also where the intensity of the image classifies a region of the road. The user must also initialize the maximum allowable distance between two points,  $\delta$ , and the minimum allowable angle,  $\theta$ , between the triplets of points. These user selected points are then interpolated, and where needed additional points are automatically added, and a B-spline is produced, corresponding to the centerline of the road. With this centerline the outer edges of the road are determined by simply copying the spline and moving it until the statistical properties of the region no longer coincide with the statistical properties of the road, designating a relative bound of the road. These newly produced edge-splines are however inaccurate and the use of region competition refines these initial estimates since widths vary and road edge paths may vary as well. These so called regions are homogeneous in such that the intensity values are similar to a pre-determined probability. In the case of roads this probability is typically Gaussian in nature since the closer one is to the center the higher the intensity and as one moves towards the edge of the road in either direction the intensity decreases. The segment of the road contour is defined by Equation 2:

$$\frac{d\vec{v}}{dt} = - \left[ \mu \kappa(\vec{v}) + \frac{1}{2} \left\{ \log \frac{\sigma^2}{\sigma_j^2} + \left( \frac{(I - \mu)^2}{\sigma^2} - \frac{(I - \mu_j)^2}{\sigma_j^2} \right) + \left( \frac{S^2}{\sigma^2} - \frac{S_j^2}{\sigma_j^2} \right) \right\} \right] \vec{n}(\vec{v}) \quad (2)$$

Where  $\kappa$  is the curvature of the contour. Each point along the contour is encompassed by a circle (region) and the value of the mean and standard deviation of the image pixels within this circular region are computed,  $I$  and  $S$  respectively. The mean and standard deviation of the road segment are defined as  $\mu$  and  $\sigma$  respectively, and the mean and standard deviation for the single

point,  $P$ , which lies on the contour, are defined as  $\mu_j$  and  $\sigma_j$  respectively. The unit normal vector to  $v$  is  $\vec{n}(\vec{v})$ . Once the statistical parameters of the point,  $P$ , are determined the likelihood ratio is computed. This determines whether the point correlates into the distribution for the road or is classified as an outlier. Figure 4 shows the referenced regions.

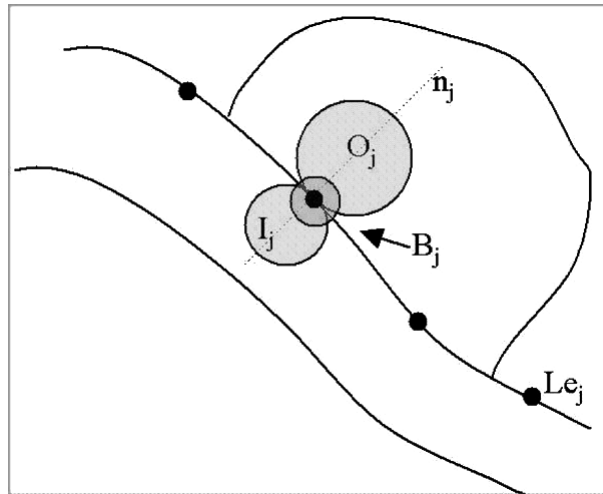


Fig. 4: Region Locations [6] pp. 1194

Gamba, Dell'Acqua, and Lisini utilize a priori information pertaining to the structure of urban road networks in combination with a Fuzzy Hough Transform (FHT) in order to extract the desired road network [7]. There are three assumptions based on a-priori information:

- 1) Urban roads align along two directions, which are not necessarily orthogonal
- 2) Roads are typically connected, and dead-end roads in an urban setting are unlikely
- 3) Road intersections are dense

The first step of this algorithm is to identify the two main directions of the road network. These directions are used in order to pre-process the image using an adaptive directional filter. The use of the FHT provides a histogram of the most probable directions of the initially extracted linear features, some of which may not necessarily be roads but most of which are. If in this histogram of potential directions, there is only one peak, then that peak is considered twice, however this case is highly unlikely. The ideal case is when the histogram contains two or more evident peaks, the two largest peaks are taken and these are the directions of the urban road network which are utilized in the directional filter. Once the image has been processed using the adaptive



directional filter, the FHT is again applied to extract more accurate estimates of roads. Some errors can arise when using the FHT, since it extracts only linear segments, curved areas require a smaller window of extraction and multiple extractions may be considered. Also portions of a road may fail to be detected due to features in the image. In this case a method based on perceptual grouping concepts is used in the final processing of the extracted roads. This process uses logical intuition to refine the road network, concepts include, continuity, collinearity, and proximity. This algorithm is governed by the selection of six main parameters defined as follows:

- 1)  $m_d$  - the maximum distance between two near parallel segments (fusion of segments, proximity concept)
- 2)  $m_e$  - the maximum distance between the extremes of a segment pair (connection of segments, continuity concept)
- 3)  $m_a$  - the maximum angle tolerance between segments (fusion, collinearity)
- 4)  $m_g$  - the maximum gap between extremes of potentially intersecting segments and the forecasted intersection point (intersection proximity)
- 5)  $m_p$  - the maximum displacement between the extracted path and its best piecewise linear approximation
- 6)  $m_l$  - the minimum length of a line segment

Amberg, Coulon, Marthon, and Spigai propose a method based on dynamic programming and the Hough Transform to extract structures from high resolution SAR data in an urban environment [8]. The first step in the proposed algorithm extracts straight lines from a local region based on the Hough Transform such that these local regions are darker in intensity when compared to neighboring regions. Then the algorithm extracts curved features, first slightly curved features and then strongly curved ones. Figure 5 outlines the steps in the algorithm.

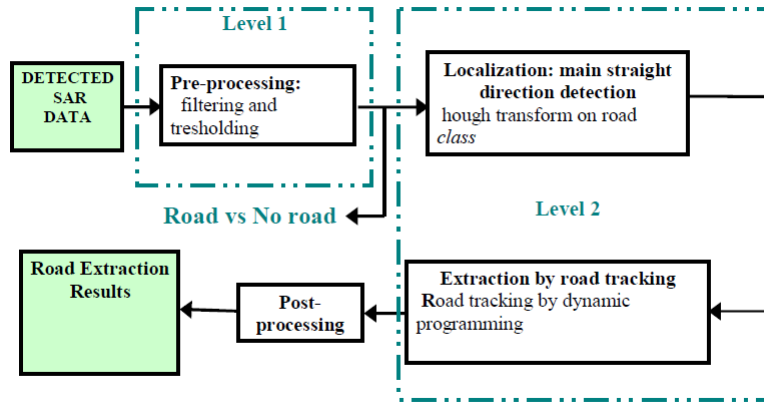


Fig. 5: Algorithm Outline [8] pp. 1785

The data is pre-processed using filters and thresholds. Then the data is grouped into road and non-road classes based on the intensity of the pixel, darker corresponding to roads. Then the large straight segments of roads are detected using the Hough Transform. The next step is to identify the segments of road subject to slight curvature. This is done using dynamic programming, which has the capability of adapting to the given data. Dynamic programming searches and adds pixels which belong to the same curve based on the minimization of the cost function. This cost function is determined by examining a tree of segments. Finally the results are smoothed with local linear approximation and gap filling with a distance criteria. The strongly curved sections of road are approximated by a hyperbolic model. The underlying assumption is that two linear segments are connected by a strongly curved segment. The hyperbolic equation is defined in Equation 3:

$$a(x-p)^2 + 2b(x-p)(y-q) + c(y-q)^2 = 1 \quad (3)$$

Where the center is defined as  $(p, q)$  and the parameters,  $(a, b, c)$ , define the shape of the hyperbola. The center of the hyperbola is estimated as the intersection point of the two linear segments it connects. As for the determination of the shape parameters, this is done using a variation of the Hough Transform. An accumulation matrix is built for a pre-defined vector of values of  $a$ ,  $b$ , and  $c$  using each pixel considered to lie on the curve. The location in this accumulation array with the highest number of votes corresponds to the best estimate of  $a$ ,  $b$ , and  $c$  based on the pre-defined vectors.

Hu, Razdan, Femiani, Cui and Wonka propose a method based on identifying a series of footprints, iteratively connecting these and then removing falsely identified paths using a Bayes decision model [9]. They identify four main concerns when attempting to extract road networks from SAR images:

- 1) Variety of roads (width, intensity, shadows) in the same image
- 2) Differing intersection models (T-shape, X-shape, Y-shape)
- 3) Variable road width due to sensor obstructions such as cars or building shadows
- 4) Image noise or physical connections between roads and surroundings causing leakage

The proposed algorithm begins with some assumptions about roads and SAR images.

- A homogeneous region around a pixel is approximately rectangular
- Roads are approximated as long thin structures with a bounded width
- In SAR images the intensity of roads are either brighter or darker than the surroundings however the intensity may not be constant due to obstructions in the sensor's path such as vehicles and shadows cast by buildings or trees

The first step is to identify the so called footprints, which are polygon regions around an identified road pixel. A road pixel, denoted by  $p$ , lies at the center of a spoke wheel. This road pixel is automatically identified by examining every pixel in the image and its neighborhood. There are 64 spokes evenly spaced between 0 and  $2\pi$ . Starting at  $p$ , the algorithm moves along each spoke and observes the intensity changes in the next set of pixels. As the distance from  $p$  grows the intensity change increases. When the difference in the intensity rises above the standard deviation along the spoke, the algorithm terminates and has reached an edge pixel. This procedure repeats on each of the 63 remaining spokes until all edge pixels have been found. These edge pixels are then connected and the road footprint for road pixel,  $p$ , has been found. In Figure 6, a total of 20 footprints have been identified from the image on the left. Footprints 0 - 12 correspond to the road while the remaining footprints correspond to buildings.

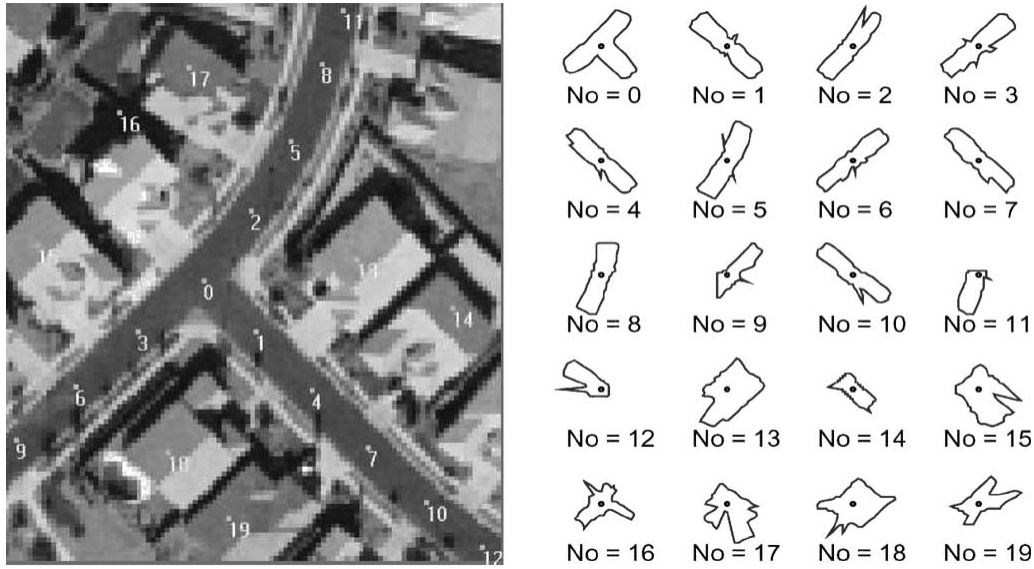


Fig. 6: Footprints [9] pp. 4147

The next phase is the application of the toe-finding algorithm. This algorithm is intended to trim the footprints and remove falsely identified branches. This algorithm is defined in the following steps:

- 1) Shift the distance function,  $\delta(i)$  so that  $\delta(0)$  is less than the average distance. Where  $i$  is an integer corresponding to the spoke number, from 0 to 64
- 2) Locate the peaks such that the local maximum distance is greater than the average distance
- 3) Remove the small peaks with respect to the maximum, and all peaks less than the average are removed. Some peak,  $j$ , is removed if the following holds:

$$\frac{\delta(j)}{\delta_{max}} < 0.25 \quad (4)$$

- 4) Merge close peaks, peaks that are within 45 degrees of one another
- 5) Remove a peak if the valley between the next peak is not deep enough, average value between two peaks,  $i_1$  and  $i_2$  is:

$$\delta_{avg} = \frac{\sum_{j=i_1}^{i_2} \delta(j)}{i_2 - i_1 + 1} \quad (5)$$

Then if the following criteria is met, the peak with the larger distance measure is chosen

and the smaller is removed

$$\frac{2\delta_{avg}}{(\delta(i_1) + \delta(i_2))} > 0.8 \quad (6)$$

The thresholds of, 0.25, 45 degrees, and 0.8 were chosen through experimentation. Figure 7 shows three examples of footprints and the toe-finding algorithm calculations of the distances. In (d) the two peaks near the end are merged since the fourth criteria in the presented algorithm is met.

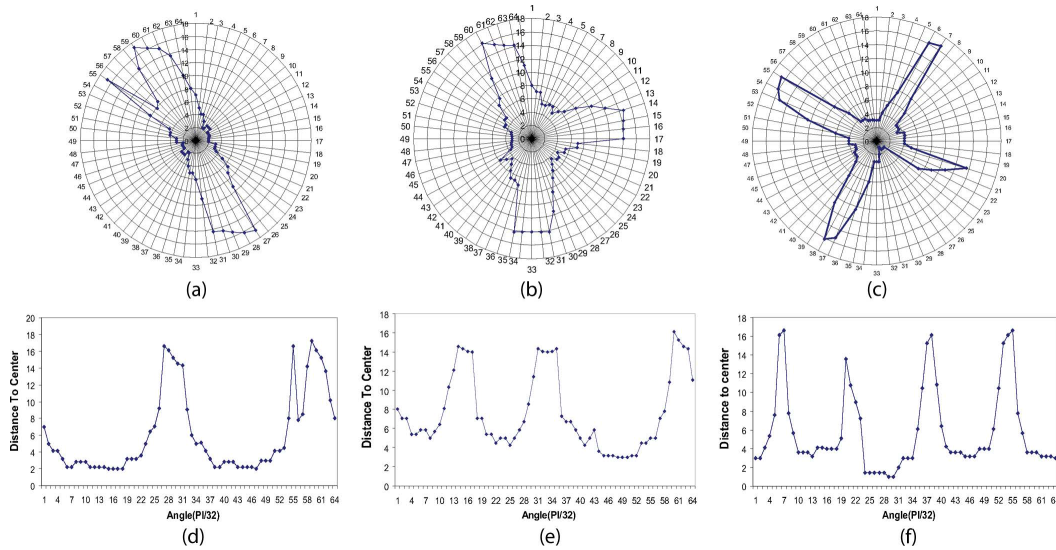


Fig. 7: Toe-Finding Algorithm [9] pp. 4148

The footprints are then classified into five groups:

- 1) Normal - two toes and the turning angle is less than 45 degrees
- 2) L-Shaped - two toes and the turning angle exceeds 45 degrees
- 3) T - Shaped - three toes
- 4) X-Shaped - four toes
- 5) Other - more than four toes

With the identified and trimmed footprints the final step of the algorithm is to remove the falsely identified footprints, referring back to Figure 6 these are the footprints labeled 13 through 19 which correspond mainly to buildings. This is done by implementing a Bayes decision rule based on the estimated width of the road at a given vertex.

Sklarz, Novoselsky, and Dorfan introduce the concept of curve to curve fusion as opposed

to the standard point to point fusion [10]. A curve is considered to be a unified sequence of points. The roadmap is a data structure consisting of nodes and road segments. These nodes are the intersections of one or more road segments. The process of accepting a track is as follows:

- 1) Track is accepted to the roadmap network
- 2) Determine whether or not the track overlaps with another existing segment
  - a) If an overlap exists, segment the new track such that the end points correspond to the overlapped track and proceed to fuse
  - b) If no overlap exists, add additional nodes and edges to the data structure

This process is shown in Figure 8 where the dashed line represents a new track.

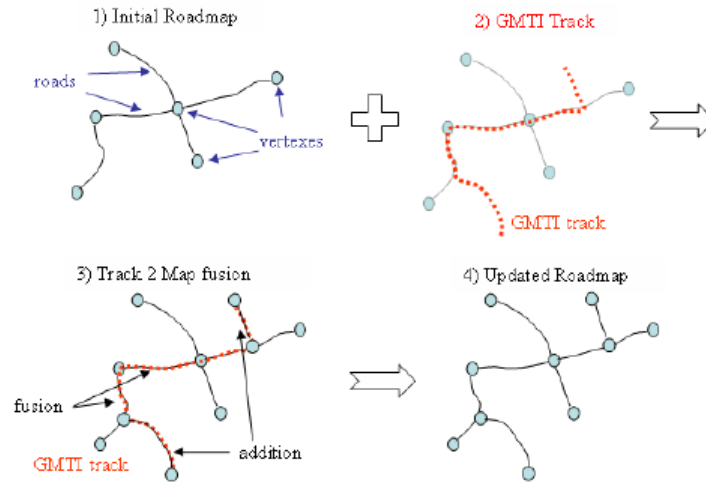


Fig. 8: New Track Processing [10] pp. 1170

In order to fuse two curve entities together a distance measure is required, in this case this is done using the inverse of the joint covariance matrix of the two curves to be fused. The dissimilarity function,  $F(\alpha(\tau))$ , is defined to be the distance between small segments of the two curves at the location denoted by,  $\tau$  along the fused curve. This function is the squared statistical distance between the two segments as shown in Equation 7:

$$F(\alpha(\tau)) = (C_1(t(\tau)) - C_2(s(\tau)))^T (P_1(t(\tau)) + P_2(s(\tau)))^{-1} (C_1(t(\tau)) - C_2(s(\tau))) \quad (7)$$

Where  $C_1(t)$  and  $C_2(s)$  denote the two curves to be fused with covariance matrices  $P_1(t)$  and  $P_2(s)$  respectively. The objective function for the optimally fused curve,  $\alpha^*$ , is defined in Equation

8:

$$\alpha^* = \arg \min_{\alpha(\tau)} \int_0^T (C_1(t(\tau)) - C_2(s(\tau)))^T (P_1(t(\tau)) + P_2(s(\tau)))^{-1} (C_1(t(\tau)) - C_2(s(\tau))) d\tau \quad (8)$$

The optimal curve,  $\alpha^*$  is obtained using dynamical programming since there are exponentially many solutions to obtain an alignment curve between two curves but only one optimal. For each point along the curve,  $(i, j)$ , the cost is computed and stored. Then it is a simple matter to search this stored data to determine the minimum. The cost at the point along the alignment curve,  $(i, j)$  is obtained from Equation 9:

$$\begin{aligned} D(i, j) = \min & \left( D(i-1, j) + \delta_{\tau(i-1, j), (i, j)} F(i, j), \right. \\ & D(i, j-1) + \delta_{\tau(i, j-1), (i, j)} F(i, j), \\ & \left. D(i-1, j-1) + \delta_{\tau(i-1, j-1), (i, j)} F(i, j) \right) \end{aligned} \quad (9)$$

Where  $\delta_{\tau_{k-1, k}} = \sqrt{(t_{i_k} - t_{i_{k-1}})^2 + (s_{j_k} - s_{j_{k-1}})^2}$ , with  $k$  and  $k-1$  composing a segment along the alignment curve.

Shackelford and Davis develop an algorithm based on the capabilities of the Hough Transform for extracting road networks from high resolution imagery [11]. The Hough Transform is explained in depth in Section 4.3.4 and is generalized here. This transformation takes a pixel, or coordinate, in the x-y space and transforms it into a  $\rho$  and  $\theta$  space. For a single coordinate the value of  $\theta$  varies over a range of -90 to 90 degrees. If a group of coordinates is shown to contribute to a line, each of these coordinates will share a point of commonality in the Hough space, some  $(\rho, \theta)$  pair, which identifies the line. Algorithms for the Hough Transform typically consist of building the Hough matrix, an  $N \times M$  matrix, corresponding to the lengths of the  $\rho$  and  $\theta$  vectors, which is an accumulation matrix of the  $(\rho, \theta)$  pairs, identifying the largest values or peaks in the accumulation matrix, and finally the ability to extract line segments and fill gaps between similar segments with the user specified inputs. The algorithm identified in [11] is based on the Hough algorithms and iterates based on the line segment size, decreasing as the iterations proceed.

## 2.2 Overview

In this report a method for developing road networks and characterizing the uncertainty in these estimates is developed. It is assumed that road networks can be broken down into two basis functions, linear and quadratic (ellipse). The initial processing of the data is done by creating a binary image and extracting possible line segments using the Hough Transform. However, the Hough Transform does not provide a measure of uncertainty, therefore the Total Least Squares approach is implemented and the Cramer Rao lower bounds is derived from the maximum likelihood estimate. The Total Least Squares solution allows for an iterative estimate which is updated in time as additional measurements become available. The Least Squares solution will be used as an initial estimate for the Total Least Squares algorithm. Once the sensor has stopped receiving data the individual line segments can be merged, extended, and blended to produce a more complete road network.

In Section 4.3.4 the Hough Transform is presented. The mathematical concepts behind the transform for the line are presented and a simple example is shown in order to understand MatLab's built in functions for the Hough Transform. Then the Hough Transform is expanded in order to identify circles and ellipses in images. The complications with both expansions are clearly explained and a few algorithms are presented for the Circle Hough transform. Finally, examples of how to reduce the parameter space for the Ellipse Hough transform are presented.

The derivations for the Least Squares, Total Least Squares, and the Least Squares ellipse are performed in Section 4.4.2. The Least Squares will be an initial estimate for the Total Least Squares algorithm. Common derivations of the Total Least Squares make assumptions about the noise associated with the measurements and the system model, however the case we are interested in involves a full fledged covariance matrix and the Total Least Squares is derived based on this assumption. The Least Squares solution for the ellipse is derived by Halir and Flusser [18] and is presented and explained here.

Section 3.11 contains the derivations involved in the uncertainty analysis for three cases. These cases include the straight line fit obtained by the Total Least Squares solution, the third order polynomial fit obtained using MatLab's *polyfit* function which solves the problem in the Least Squares sense, and finally the ellipse obtained by the Least Squares algorithm. Monte Carlo simulations and examples are performed in order to show the convergence of the derived



solutions.

In Section 4.0 the Graphical User Interface is thoroughly explained and depicted. Each function is explained and small examples of important working features are shown using the first available data set. The required format of the input data as well as the format of the output data is outlined in detail as well.

The resulting interface is utilized along with two available data sets and in Section 5.0 the results are shown. Since the data was originally in Latitude and Longitude coordinates it is prudent to obtain the final results in the same coordinate system. Results within the GUI are shown along with results of externally utilizing the output data structure. The numerical results of the Cramer Rao lower bounds computations performed in Section 3.11 are presented and explained for each of the provided data sets.

In Section 5.0 we make our concluding remarks about the work done throughout this report and outline some potential future work dealing with the implementation as well as additional features which may be incorporated at the discretion of the user.

## 3.0 METHODS

### 3.1 Hough Transform

The Hough Transform was first introduced in 1962 by Paul Hough and was generalized in 1971 by Richard Duda and Peter Hart [12] and is utilized to determine geometric features in binary images. The most general transformation is used to detect lines but the concept can be further developed to detect circles [ [12], [13]] and ellipses [ [13], [14], [15]] in images. This transformation uses a voting procedure in which the algorithm takes the binary pixel's coordinates and computes the parameters required by the geometric feature. The parameters necessary vary for each feature, a line requires two parameters, while the circle three and the ellipse five. Thus the accumulation array increases in dimension and the algorithm's computational complexity drastically increases in proportion to the number of parameters. The Hough Transform for the line, circle, and ellipse are explored in the next sections.

### 3.2 Line Hough Transform

The Hough Transform eliminates the complications which arise with the identification of vertical lines in an image. This transform requires the image to be binary in nature, where white pixels correspond to ones and black pixels correspond to zeros. The derivation of the Hough Transform requires basic trigonometric properties. Suppose we have a line oriented as shown in Figure 9 then by defining the parameters,  $\rho$ , and  $\theta$  we can derive the Hough Transform. The perpendicular distance from the origin to a line, is denoted by  $\rho$ . The angle that this distance vector makes with the x-axis is  $\theta$ .

We note the definitions of the cosine and sine functions:

$$\cos \theta = \frac{x}{\rho} \quad \sin \theta = \frac{y}{\rho} \quad (10)$$

Algebraic substitution of a single cosine and sine term in the Pythagorean Theorem:

$$\frac{x}{\rho} \cos \theta + \frac{y}{\rho} \sin \theta = 1 \quad (11)$$

It is now a simple matter to rearrange Equation 11 to obtain the conventional form of the Hough Transform as given by Equation 12.

$$\rho = x \cos \theta + y \sin \theta \quad (12)$$

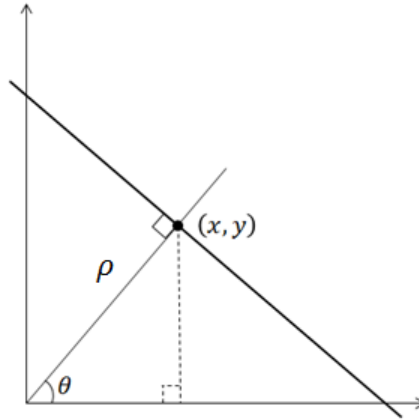


Fig. 9: Parameter Identification

Now if Equation 12 is rearranged into a slope-intercept form we can infer that when  $\theta$  approaches zero degrees, corresponding to a vertical line, the same issue occurs such that the slope tends to infinity. However, using Equation 12 alleviates this problem.

$$y = -\frac{\cos \theta}{\sin \theta}x + \frac{r}{\sin \theta} \quad (13)$$

The principle concept of the Hough Transform in the line identification algorithm can be stated as the following: **if two points are collinear then they share a pair,  $(\rho, \theta)$  of commonality in the Hough space.** However, in order to determine this common pair, a Hough matrix must be constructed, this is done by iterating over a  $\theta$  range of -90 to 90 degrees for each white pixel, which corresponds to the  $(x, y)$  coordinates, in the image. Recall that  $\theta$  is the angle the  $\rho$  vector makes with the x-axis. We can imagine that an arbitrary number of lines pass through each coordinate at the discretized values of  $\theta$  such as in Figure 10. The figure shows two collinear points with the same discretization of  $\theta$ . The closest pair of  $(\rho, \theta)$  is shown. Each of the dashed lines in Figure 10 represents the perpendicular distance from the origin to the solid line where they terminate (two of the distance vectors are hidden from view due to the axes). Since  $\rho$  in the Hough Transform is contained in a finite length vector, the nearest discrete value of  $\rho$ , within the vector, must be determined. This is done using a simple linear transformation as shown in Figure 11.

Thus to determine the appropriate *RIdx* value Equation 14 is utilized, one is added depending

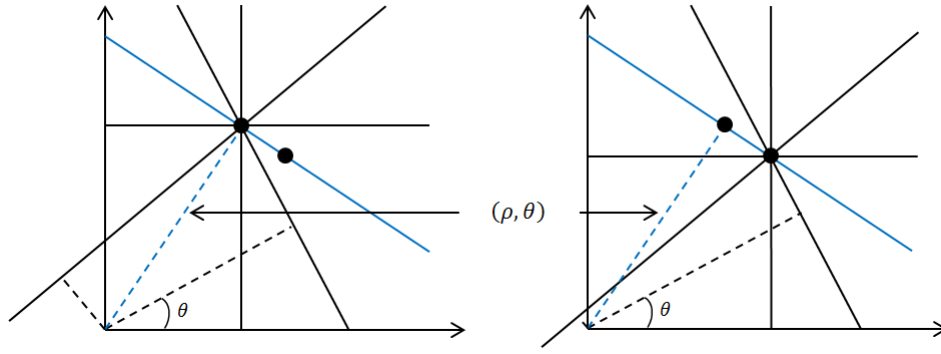


Fig. 10: Collinear Points

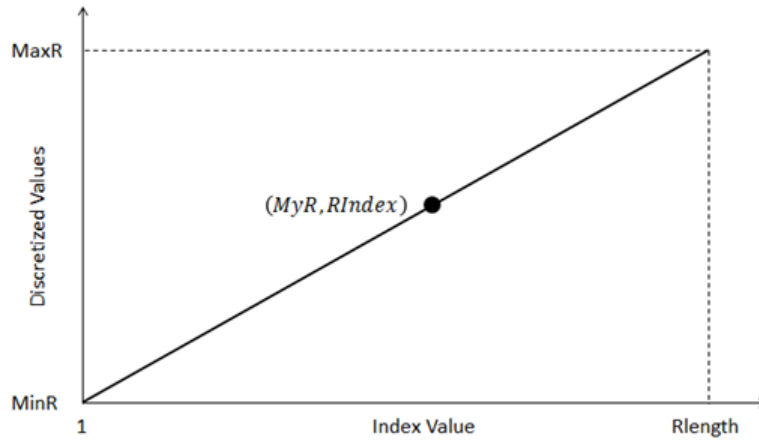


Fig. 11: Linear Transformation of  $\rho$

on the first index utilized in the corresponding language (MatLab is utilized here).

$$RIIdx = \text{round} \left( \frac{Rlength - 1}{MaxR - MinR} (MyR - MinR) \right) + 1 \quad (14)$$

For each point in the image, we must iterate over the set  $\theta$  range and determine the corresponding  $RIIdx$  value for each case. The corresponding index pair,  $(\rho, \theta)$  in the Hough matrix is then incremented and this process continues until each pixel's Hough Transform has been computed. For each pixel in the image with a coordinate  $(x, y)$ , the value of  $\rho$  is plotted for the set of discrete values of  $\theta$ . This is repeated for all pixels to generate an image such as the one shown in Figure 12. The common point in the Hough space is highlighted in the figure which corresponds

to the intersection of the sinusoidal curves.

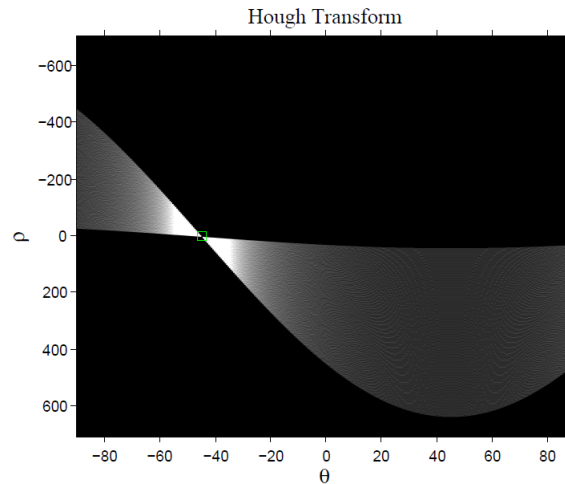


Fig. 12: Hough Transform

MatLab has a set of built in functions which can be used to extract lines from the Hough matrix. These functions consist of:

- *hough*, creates the Hough matrix, options:
  - *ThetaResolution* - specifies the discretization of the  $\theta$  parameter
  - *RhoResolution* - specifies the discretization of the  $\rho$  parameter
- *houghpeaks*, determines the  $(\rho, \theta)$  pairs with the highest number of votes, options:
  - *N* - specifies the number of peaks to identify (only peaks satisfying *Threshold* will be extracted)
  - *Threshold* - minimum number of collinear points to extract a line
  - *NHoodSize* - the region around the identified peak which is eliminated upon identification
- *houghlines*, uses the peaks to determine the associated line segments, returns a data structure containing endpoints of line segments, options:
  - *FillGap* - distance between two line segments which will be filled in order to bridge gaps
  - *MinLength* - minimum allowable length of a line segment

Once the endpoints and the  $(\rho, \theta)$  pair have been identified, the corresponding data points can be associated with the line estimate; this builds the data clusters for the next stage.

### 3.3 Circle Hough Transform

In the line Hough there were two unknown parameters which defined the line. However with the circle there are a total of three parameters, the center coordinates,  $(x_0, y_0)$  and the radius,  $R$ . The generic equation for a circle is given to be:

$$(x - x_0)^2 + (y - y_0)^2 = R^2 \quad (15)$$

Figure 13 shows a circle with the parameters identified. Suppose a point,  $(x_i, y_i)$  lies on the circle

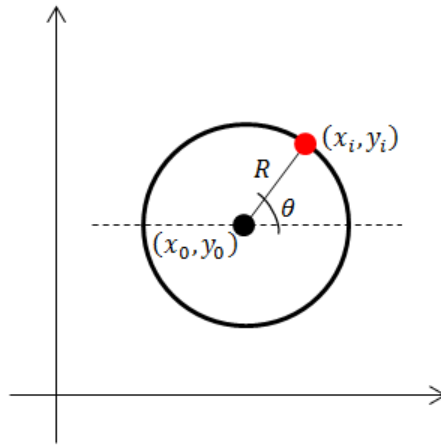


Fig. 13: Circle Parameter Identification

described by the center,  $(x_0, y_0)$  and the radius,  $R$ , we can then determine the corresponding values of the center coordinates assuming that the radius of the circle is known. The angle,  $\theta$  is the angle between the arbitrary x-axis located at  $x_0$  and the point lying on the edge of the circle. Using basic trigonometry the equations for the center coordinates are as follows:

$$x_0 = x_i - R \cos \theta \quad (16)$$

$$y_0 = y_i - R \sin \theta \quad (17)$$

Since we assume that  $R$  is known, the accumulation matrix is built in the same manner as with the line Hough Transform, with the variation that  $\theta$  now ranges from 0 to 360 degrees. Therefore,

for each point in the image, the center coordinates are computed based on the known value of the radius and the discretized values of  $\theta$ . However, prior knowledge of the radius is unlikely and therefore the Circle Hough Transform (CHT) becomes more complex. It is therefore necessary to establish a vector of radius values (which may be based on the image size) which can be used to search for the optimally fit circle. In this manner for each  $R_j$  the center coordinates are computed with the discretized values of  $\theta$ . The construction of the accumulation matrix is summarized as follows:

- Iterate over each value of  $R_j$  (dimension 3)
- Next iterate over each coordinate pixel,  $(x_i, y_i)$
- Finally iterate through the discretized  $\theta$  vector and compute the center coordinates,  $(x_0, y_0)$  (dimensions 1 and 2)

The accumulation matrix is a three dimensional matrix as shown in Figure 14. MatLab does not

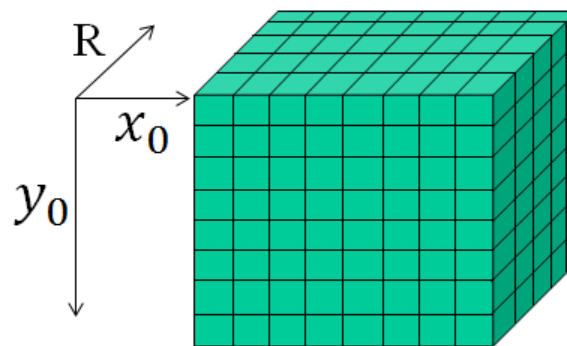


Fig. 14: Accumulation Matrix

contain any built in functions for the CHT however Tao Peng has developed an algorithm which solves many of the complications which arise in images such as overlapping circles, incomplete circles, unknown radii, multiple circles, and tightly spaced circles. However we begin with our own simple code to illustrate the concept and present results from Peng's work later.

Using Equations 16 and 17 it is possible to construct a basic algorithm to illustrate the CHT. In reality the code can be generalized to accept any initial input based on the user's design, it need not be a binary image, however to keep with the Hough Transform's convention we will establish the input to be a binary image. We assume unequal and unknown radii for two circles

as shown in Figure 15.

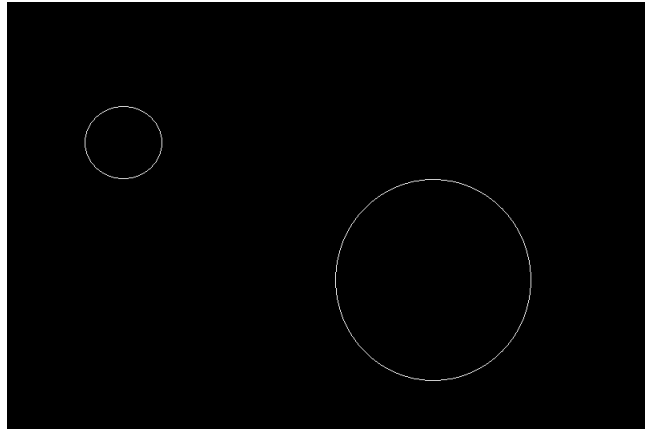


Fig. 15: Unequal Non-concentric Circles

To begin with, the ranges need to be specified on the radii as well as the potential location of the center coordinates. Vectors of the center coordinates are established to create indexing for the accumulator matrix. With the vectors specified, the computed values of the center coordinates are then rounded to the nearest index value in the same manner as the  $RIdx$  value in the linear Hough Transform. Then as outlined previously, we must iterate over the range of radius values, and then each coordinate pixel, and finally each value of  $\theta$ . This iteration procedure builds the 3-dimensional Hough matrix which contains the votes for the parameter triplet,  $(R_j, x_0, y_0)$ . Once the Hough matrix is built, we can then search this matrix for the highest number of votes and attribute that 3-dimensional location to the parameters which define a circle. Since we developed this simplified version of the CHT algorithm with the assumption that the circles have unequal radii, we simply employ the *houghpeaks* function after each 2-dimensional matrix is composed, which then provides the peak associated with each radius,  $R_j$ . These peaks are then stored into a row vector with the same length as the radius vector and when complete, we simply search through this vector for the two highest values corresponding to the appropriate radius values. This algorithm is summarized below:

- 1) Initialize radius and center coordinate vectors
- 2) Set the number of circles in the image (assumed known)
- 3) Iterate over the radius vector



- 4) For each radius, iterate over each pixel
- 5) For each pixel, iterate over the  $\theta$  range, 0 to 360 degrees and compute the center coordinates
- 6) Determine the nearest integer index value for the computed coordinates
- 7) For each radius, using *houghpeaks*, extract the single highest peak, corresponding to the center coordinates at each radius value
- 8) Search through the candidate peaks vector, from previous step, and locate the highest  $N$  counts, where  $N$  is the number of circles. This corresponds to the radii.
- 9) Using the corresponding radius index value, return to the candidate peaks vector and locate the center coordinates

The results of the simple CHT algorithm applied to the image shown in Figure 15 is shown in Figure 16.

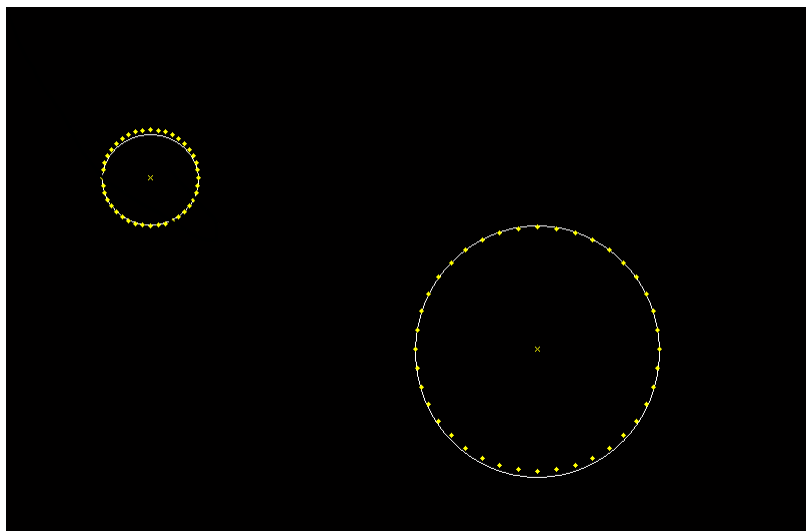


Fig. 16: Simple CHT Algorithm Results

As can be seen from Figure 16 the center coordinates of the circles are off slightly. This is due to the discretization and rounding in the indexing of the computed values. Table I shows the actual values (by using the MatLab data cursor) and the CHT values for the radii and the center coordinates. The first circle corresponds to the smaller of the two and the second is the larger.

TABLE I: CHT Numerical Comparison

<b>Circle 1</b>	$R$	$x_0$	$y_0$	<b>Circle 2</b>	$R$	$x_0$	$y_0$
Actual	48	146	176	Actual	122	533	347
CHT	48	146	175	CHT	122	533	345

With this simplified algorithm for the CHT there are many drawbacks. First, a minor issue is that the user must specify the range for the center coordinates, this is intended to save space in initializing the Hough matrix. The next problem is the prior knowledge of the number of circles in the image, and then the ability to properly identify these circles. The identification is subject to several potential scenarios which were previously described such as, overlaps, incomplete circles in the image, and tightly spaced circles (the potential to identify a circle between the two actual circles due to the voting).

Now we will take a look at Tao Peng's algorithm and its capabilities at handling the issues identified. This algorithm can be found via the MathWorks website and has the following inputs:

- *img* - a grey scale image (should be altered before input)
- *radrange* - the range of radius values which the algorithm will search over
- *grdthres* - gradient magnitude threshold, used to remove the background image
- *fltr4LM\_R* - filter radius used in the searching of maximum values in the accumulation array
- *multirad* - parameter which allows for the detection of multiple concentric circles
- *fltr4accum* - filter to smooth the accumulation array, similar to standard MatLab filters

Also the outputs of the algorithm are:

- *accum* - accumulation array from the CHT
- *circen* - centers of all detected circles in the image
- *cirrad* - radius values of all detected circles
- *dbg\_LMmask* - outputs the areas of interest in detecting the local maxima in the accumulation matrix

The algorithm first scans through the image and removes all background pixels. Since this algorithm requires a grey scale image as opposed to the standard binary, the gradient magnitude will determine the regions of higher intensity, corresponding to brighter pixels and these will be considered as the desired data points for the building of the Hough matrix. Once this separation has been achieved, the filter is applied to smooth the circle borders if needed and also to thin the amount of pixels corresponding to a circle. For example if the circle is completely filled in with white pixels, the filtering allows a form of edge detection in which higher weights are attributed to the points which lie on an edge and the inner points are reduced. The remainder of the algorithm progresses in the same manner as the simplified version. The accumulation

array for the centers is built and then searched for local maxima such that the value exceeds the threshold. Once the centers have been obtained the corresponding data points which produced the votes for this center are then separated and the accumulation array is built for the radius values. The multiple radius detection is possible in this step with a proper thresholding. This radius accumulation array for each center point (previously identified) is then searched for the one or more local maxima depending on the *multirad* parameter.

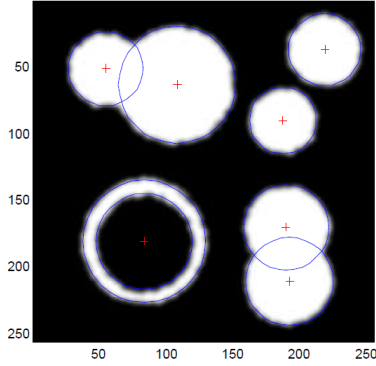
We now present an example from the supplied package. Figure 17 shows the grey scale image which contains overlapping circles and concentric circles.



Fig. 17: Tao Peng Example 2

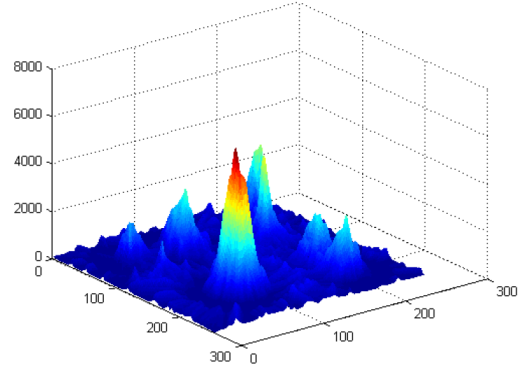
The algorithm is then applied and we can see the 3-dimensional accumulation array and the produced results in Figure 18

Raw Image with Circles Detected (center positions and radii marked)



(a) CHT Results

3-D View of the Accumulation Array



(b) Accumulation Array

Fig. 18: Results from Tao Peng's CHT Algorithm

Another algorithm has been developed by David Young which is much simpler in its implementation when compared to Tao Peng and has similar results. In Young's algorithm he saves computation time by creating a 1-dimensional array whose length is  $N_x \times N_y \times N_r$  (multiplication not matrix dimensions), where  $N$  refers to the search array of the  $x$ ,  $y$ , and  $r$  spaces respectively. The complete CHT is separated into three functions defined below:

- *circle\_hough* - this function takes in the image, and the radius range and outputs the Hough matrix which in the end is a 3-dimensional matrix
- *circle\_houghpeaks* - locates the local maxima in the hough matrix, requires inputs of the Hough matrix and the radius range. Also optional inputs include the suppression neighborhood for the center and the radius, as well as the number of maxima to extract
- *circlepoints* - creates a template of points for each radius value, centered at zero in order to build the 1-dimensional array Hough

This algorithm is able to detect multiple concentric circles as well as overlapping circles and requires roughly the same computational time as Peng's algorithm however it is much simpler in its implementation. We show here the results on the same image from Figure 17, the green circles are the identified circles by Young's CHT algorithm.

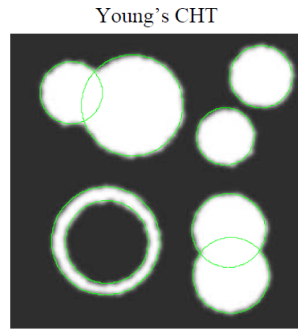


Fig. 19: Results from David Young's CHT Algorithm

Guil and Zapata utilize a Fast Hough Transform (FaHT) in their detection algorithms for both the circle and ellipse [13]. The FaHT essentially uses a coarse quantization of the initial parameter space in order to locally identify regions of interest with a high number of votes in the accumulation array. After these regions are identified a finer level of quantization is locally implemented and searched again for the local maxima in the accumulation array. This speeds up the algorithm since it no longer searches the entire parameter space defined with the fine level of quantization. The CHT algorithm is broken up into two stages. In the first stage the candidate centers are located by observing the intersection of the pixel's gradient vector. The points considered in these candidate centers are then stored to be analyzed for the potential radius values. In the second stage the radius is determined, the distance from the candidate center to each stored pixel is computed and the corresponding index in the accumulation array is incremented.

### 3.4 Ellipse Hough Transform

The Hough Transform can be further expanded to detect elliptical shapes in images. However as seen in the CHT the parameter space increased from the traditional 2-dimensional space to a 3- dimensional space thus exponentially increasing the computational requirements. In the Ellipse Hough Transform (EHT) the dimension now increases to a 5-dimensional parameter space defined by, the center coordinates,  $(x_0, y_0)$ , the major and minor axis,  $a'$  and  $b'$  respectively, and finally the rotational angle of the ellipse,  $\phi$ . The computational complexity of the standard EHT algorithm has now increased exponentially as an accumulator array of dimension five must be built and searched for the optimal set of parameters. Therefore it is common in practice to reduce this space as much as possible. Several authors have separated the accumulator array into two arrays, the first being 2-dimensional and containing the potential center coordinates and the remaining 3-dimensional array contains the major and minor axis as well as the rotational angle of the ellipse.

Tsuji and Matsumoto [14] first eliminate long linear segments using the traditional Hough Transform. Then from the remaining line segments, they search these candidates to find parallel segments. The edge points of these segments are then averaged and the 2-dimensional accumulator array is then incremented. Once all the line segments which failed to meet the minimum length criteria have been processed in this manner, the potential center locations are extracted from the 2-dimensional array. The edge points are then stored in a matrix, however these edge points need not lie exactly on an ellipse, therefore these candidate edge points are then processed to determine if they do in fact lie within an acceptable distance to the ellipse.

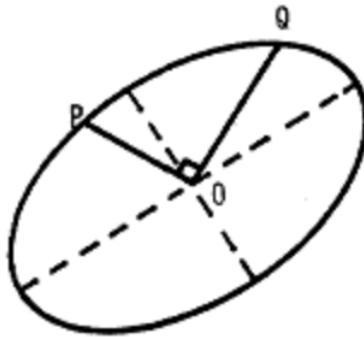


Fig. 20: Candidate Ellipse Points [14] pp. 778

This is done by considering the angle between two points as shown in Figure 20. If this angle is  $90^\circ \pm \delta$  where  $\delta$  is some tolerance, then the pair of points lies on the ellipse. Once all pairs of candidate edge points have been processed in this manner, a least mean squares algorithm is applied to determine the five parameters. The initial stages of the algorithm were intended only to identify the data points which correspond to each ellipse and separate them into their own structures.

Aguado and Nixon [15] reduce the parameter space by utilizing gradient direction and tangential vectors. Two tangential vectors are used to determine the center of the ellipse. Extrapolating these vectors to their intersection point,  $T$ , and finding the midpoint,  $M$ , between the two points which lie on the ellipse,  $P_1$  and  $P_2$ , then by extrapolating the line,  $TM$  one can locate the ellipse's center as shown in Figure 21.

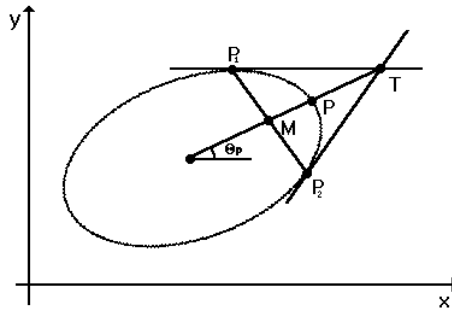


Fig. 21: Ellipse Center Location [15]

Using gradient information, the relation between the ellipse center and the angular information is given by:

$$\frac{y''}{x''} = \frac{y - b_0}{x - a_0} \quad (18)$$

The orthonormality property of the axes gives us the second necessary equation:

$$\tan(\phi_1 - \rho) \tan(\phi_2 - \rho) = N^2 \quad (19)$$

With the following parameters defined:

$$\phi_1 = \tan^{-1} \left( \frac{y'}{x'} \right) \quad \phi_2 = \tan^{-1} \left( \frac{y''}{x''} \right) \quad \rho = \tan^{-1}(K) \quad K = \frac{a_y}{a_x} \quad N = \frac{b_y}{a_x}$$



Where ' denotes the first derivative and '' denotes the second derivative of the respective parameter. Therefore rather than a 5-dimensional parameter space the EHT algorithm has been separated into two, 2-dimensional spaces. The first 2-dimensional space builds the accumulator array for the center coordinates,  $x_0, y_0$  and the second 2-dimensional space builds the accumulator for the axes ratios,  $K$  and  $N$ .

In [13] the EHT parameter space is simplified into two 2-dimensional accumulation arrays. The first of which is used to identify the center coordinates in the same method as in [15], each pixel's gradient vector is extrapolated and the intersection of a multitude of gradient vectors defines a candidate center location. Rather than searching for the remaining three parameters, the major and minor axes and the orientation angle, the axes ratio is utilized to reduce the secondary parameter space to a 2-dimensional space. The EHT algorithm presented in [13] is also based on the Fast Hough Transform in which the axes ratio and orientation angle are searched on a coarse quantization level and then the local maxima are obtained and a finer resolution area is searched to refine the maxima. The orientation angle and axes ratio,  $\phi$  and  $h$  respectively, must satisfy Equation 20. Where the axes ratio is,  $h = a^2/b^2$ :

$$h = \frac{(x - x_0) \cos \phi + (y - y_0) \sin \phi}{(x - x_0) \sin \phi + (y - y_0) \cos \phi} \tan(\zeta - \phi) \quad (20)$$

The Fast EHT algorithm in stage two searches the parameter space for the orientation angle and axes ratio with the known center coordinates,  $(x_0, y_0)$ , for a given pixel in the image,  $(x, y)$ , with the angle of its gradient vector,  $\zeta$ . The axes ratio can be transformed using the following set of equations:

$$b = \sqrt{h^{-1}(x - x_0)^2 - (y - y_0)^2} \quad (21)$$

$$a = \sqrt{(x - x_0)^2 + h(y - y_0)^2} \quad (22)$$

In what follows we will define a method based on the Least Squares solution of fitting an ellipse to a set of data points [18]. This solution is explored in the next section and supplemented with examples. The downside to this implementation is that it assumes that all data points passed into the algorithm correspond to a single ellipse.

### 3.5 Maximum Likelihood Estimators

In this section we will derive the solutions for the standard Least Squares, a general form of the Total Least Squares to allow for a fully populated covariance matrix, and finally the Least Squares solution for the ellipse fitting problem as defined in [17] and [18]. The Hough Transform in the Section 4.3.4 will be utilized to identify groups of nearly collinear points in an image. A best fit line can be obtained using the Total Least Squares(TLS) solution since there is noise in both the  $x$  and  $y$  directions. The TLS solution which will be derived does not have a closed form solution and therefore it requires an initial estimate to solve for a better approximation to the linear coefficients. This initial estimate can be given by a transformed version of the Hough coefficients (i.e. transform from  $\rho$  and  $\theta$  to slope,  $m$  and intercept,  $b$ ), however if  $\theta$  is zero degrees then we would obtain an infinite slope, which forces the TLS solution to diverge. Therefore, rather than use the transformed Hough Transform coefficients, we will simply utilize a Least Squares estimate as the initial guess for the TLS algorithm.

### 3.6 Least Squares

In this section we derive the Least Squares solution. The least squares fit for a data set minimizes the vertical distance between the curve fit and the measurements, thus assuming that there is noise in the observation. Figure 22 shows a group of data points with a line fit, where the minimization of the distance between the measurements and the line in the vertical direction is the objective of the Least Squares method. In order to minimize the distance we begin by establishing the error function which is a simple summation of vertical distances over each of the measurements,  $M$ . A generalized version of the measurement equation is given by:

$$\mathbf{y} = \mathbf{H}\mathbf{x} + \nu \quad (23)$$

Where  $\mathbf{H}$  is the  $M \times N$  matrix of measurements,  $\mathbf{y}$  is a  $M \times 1$  vector of observations,  $\mathbf{x}$  is a  $N \times 1$  vector composed of the coefficients of the model fit, whether it be linear, quadratic, cubic, or a higher order fit and the observation noise is given by  $\nu$ . The generalized form gives the following error function:

$$E = (\mathbf{H}\mathbf{x} - \mathbf{y})^T \mathbf{R}^{-1} (\mathbf{H}\mathbf{x} - \mathbf{y}) \quad (24)$$

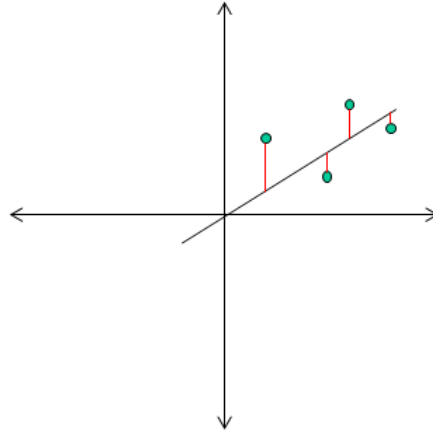


Fig. 22: Least Squares Minimization

Where  $\mathbf{R}$ ,  $M \times M$ , denotes the covariance matrix of the observation noise. We then need to expand the error function so that we can take the derivative to obtain the optimal solution.

$$E = \mathbf{x}^T \mathbf{H}^T \mathbf{R}^{-1} \mathbf{H} \mathbf{x} - 2 \mathbf{x}^T \mathbf{H}^T \mathbf{R}^{-1} \mathbf{y} + \mathbf{y}^T \mathbf{R}^{-1} \mathbf{y} \quad (25)$$

Next we compute the derivative of the expanded error function with respect to  $\mathbf{x}^T$  and set it equal to zero to obtain the optimal solution.

$$\frac{dE}{d\mathbf{x}^T} = 2\mathbf{H}^T \mathbf{R}^{-1} \mathbf{H} \mathbf{x} - 2\mathbf{H}^T \mathbf{R}^{-1} \mathbf{y} = 0 \quad (26)$$

Therefore, the solution for the estimated coefficients of the data fit is given by Equation 27.

$$\hat{\mathbf{x}} = (\mathbf{H}^T \mathbf{R}^{-1} \mathbf{H})^{-1} \mathbf{H}^T \mathbf{R}^{-1} \mathbf{y} \quad (27)$$

This is the standard least squares solution used in most regression models, where  $\hat{\mathbf{x}}$  denotes the vector of the coefficient estimates.

### 3.7 Total Least Squares

In this section we will derive the Total Least Squares solution, where the difference is that the Total Least Squares uses a model which contains error in both the  $\mathbf{H}$  and  $\mathbf{y}$  terms and therefore seeks to minimize the Mahalanbois distance between the measurements and the curve fit. For the case of a simple linear regression, in which the noise in  $x$  and  $y$  are equal, Figure 23 is shown, which displays the measurements and the distance to be minimized (in this special case

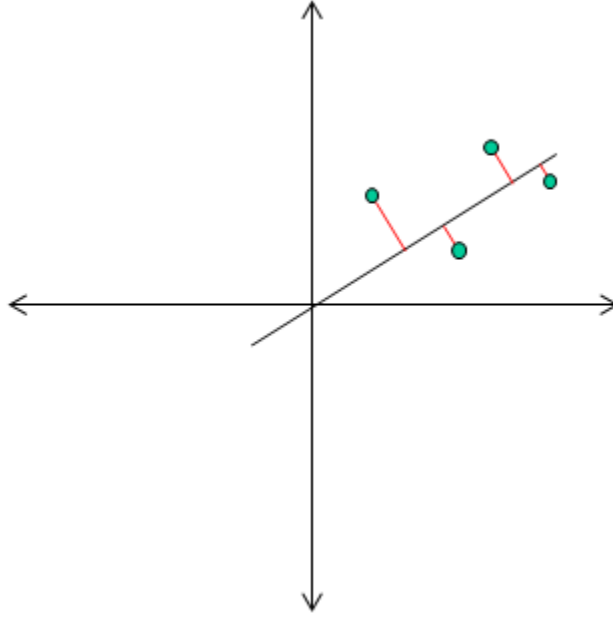


Fig. 23: Total Least Squares Minimization

the distance is perpendicular).

The derivation we are interested is presented in [16], for the case of uncorrelated non-stationary noise. The noise is allowed to vary in time, however, they are independent in time. In other words, the measurements are independent with varying covariance matrices. The model is given by:

$$\tilde{\mathbf{y}} = \mathbf{y} + \nu_y \quad (28)$$

$$\tilde{\mathbf{H}} = \mathbf{H} + \nu_H \quad (29)$$

Where  $\nu_y$  and  $\nu_H$  are the noise in the observations and measurements respectively. The first problem here is to determine the suitable estimates for  $\mathbf{y}$  and  $\mathbf{H}$  since the known values are,  $\tilde{\mathbf{y}}$  and  $\tilde{\mathbf{H}}$ . The vector of known quantities is established as,  $\tilde{\mathbf{D}} = [\tilde{\mathbf{H}} \tilde{\mathbf{y}}]$  and the vector of estimated values,  $\hat{\mathbf{D}} = [\hat{\mathbf{H}} \hat{\mathbf{y}}]$ . Thus the probability of  $\tilde{\mathbf{D}}$  given  $\hat{\mathbf{D}}$  is given to be:

$$p(\tilde{\mathbf{D}}|\hat{\mathbf{D}}) = \frac{1}{(2\pi)^{M/2} \sqrt{|\mathbf{R}|}} e^{\left(-\frac{1}{2}(\tilde{\mathbf{D}}^T - \hat{\mathbf{D}}^T)^T \mathbf{R}^{-1} (\tilde{\mathbf{D}}^T - \hat{\mathbf{D}}^T)\right)} \quad (30)$$

Where  $|\mathbf{R}|$  denotes the determinant of the covariance matrix. Since the measurements are assumed to be independent the likelihood function can be rewritten as a product of the individual

measurement PDFs. Taking the negative of the log-likelihood function establishes the objective function to be maximized, however the objective function is constrained as follows:

$$\arg \min_{\hat{\mathbf{d}}_i} J(\hat{\mathbf{x}}) = \frac{1}{2} \sum_{i=1}^M \left( \tilde{\mathbf{d}}_i - \hat{\mathbf{d}}_i \right)^T \mathbf{R}_i^{-1} \left( \tilde{\mathbf{d}}_i - \hat{\mathbf{d}}_i \right) \quad (31)$$

$$\text{Subject to} \quad \hat{\mathbf{d}}_i^T \hat{\mathbf{z}} = 0 \quad (32)$$

Since the measurements are independent of each other, the objective function becomes a summation due to the properties of the log function. The vector  $\hat{\mathbf{z}}$  is a row vector defined as  $[\hat{\mathbf{x}}^T, -1]^T$  and  $\hat{\mathbf{d}}_i$  and  $\tilde{\mathbf{d}}_i$  are the  $i$ th row of their respective matrices, transposed, giving a  $N+1 \times 1$  vector. We must now define the covariance matrix,  $\mathbf{R}$ . For each measurement we have stated that these matrices are independent and unequal. For a single measurement,  $i$ , the covariance matrix is:

$$\mathbf{R}_i = \begin{bmatrix} \mathbf{R}_{h_i h_i} & \mathbf{R}_{h_i y_i} \\ \mathbf{R}_{h_i y_i}^T & R_{y_i y_i} \end{bmatrix} \quad (33)$$

This problem needs to be reformulated as an unconstrained optimization problem. This can be done with the use of Lagrange multipliers, which requires the constraint function to be multiplied by a scalar parameter,  $\lambda_i$ . There are  $M$  Lagrange multipliers, one for each constraint. The unconstrained objective function is then modified by adding the Lagrange multiplied constraints:

$$J(\hat{\mathbf{x}}) = \lambda_1 \tilde{\mathbf{d}}_1^T \hat{\mathbf{z}} + \lambda_2 \tilde{\mathbf{d}}_2^T \hat{\mathbf{z}} + \dots + \lambda_M \tilde{\mathbf{d}}_M^T \hat{\mathbf{z}} + \frac{1}{2} \sum_{i=1}^M \left( \tilde{\mathbf{d}}_i - \hat{\mathbf{d}}_i \right)^T \mathbf{R}_i^{-1} \left( \tilde{\mathbf{d}}_i - \hat{\mathbf{d}}_i \right) \quad (34)$$

For a single measurement,  $i$ , the expanded objective function yields:

$$J(\hat{\mathbf{x}}) = \lambda_i \tilde{\mathbf{d}}_i^T \hat{\mathbf{z}} + \frac{1}{2} \tilde{\mathbf{d}}_i^T \mathbf{R}_i^{-1} \tilde{\mathbf{d}}_i - \hat{\mathbf{d}}_i^T \mathbf{R}_i^{-1} \tilde{\mathbf{d}}_i + \frac{1}{2} \hat{\mathbf{d}}_i^T \mathbf{R}_i^{-1} \hat{\mathbf{d}}_i \quad (35)$$

Our first concern is to determine the optimal solution for,  $\hat{\mathbf{d}}_i$ . In order to so, we take the derivative with respect to  $\hat{\mathbf{d}}_i^T$  and solve for the Lagrange multiplier.

$$\frac{dJ(\hat{\mathbf{x}})}{d\hat{\mathbf{d}}_i^T} = \lambda_i \hat{\mathbf{z}} - \mathbf{R}_i^{-1} \tilde{\mathbf{d}}_i + \mathbf{R}_i^{-1} \hat{\mathbf{d}}_i = 0 \quad (36)$$

To remove the inverse covariance matrix, left multiply by,  $\hat{\mathbf{z}}^T \mathbf{R}_i$ :

$$\hat{\mathbf{z}}^T \mathbf{R}_i \lambda_i \hat{\mathbf{z}} = \hat{\mathbf{z}}^T \tilde{\mathbf{d}}_i - \hat{\mathbf{z}}^T \hat{\mathbf{d}}_i \quad (37)$$

Recalling the constraint,  $\hat{\mathbf{d}}_i^T \hat{\mathbf{z}} = 0$  and that the Lagrange multiplier,  $\lambda_i$ , is a scalar quantity, it is straightforward to solve for the Lagrange multiplier:

$$\lambda_i = \frac{\hat{\mathbf{z}}^T \tilde{\mathbf{d}}_i}{\hat{\mathbf{z}}^T \mathbf{R}_i \hat{\mathbf{z}}} \quad (38)$$

Since the multiplication,  $\hat{\mathbf{z}}^T \mathbf{R}_i \hat{\mathbf{z}}$ , will always produce a scalar quantity we can simply divide this term through. Substituting Equation 38 into 36 to obtain the optimal solution for  $\hat{\mathbf{d}}_i$ :

$$\hat{\mathbf{d}}_i^T = \tilde{\mathbf{d}}_i^T - \frac{\mathbf{R}_i \hat{\mathbf{z}}^T \tilde{\mathbf{d}}_i \hat{\mathbf{z}}}{\hat{\mathbf{z}}^T \mathbf{R}_i \hat{\mathbf{z}}} \quad (39)$$

We will denote  $\tilde{\mathbf{d}}_i^T \hat{\mathbf{z}}$  as  $e_i$  since if performing the vector multiplication we see that this is in fact the error in the measurement,  $\tilde{\mathbf{h}}_i^T \hat{\mathbf{x}} - \tilde{y}_i$ . We can perform the matrix-vector multiplication and obtain separate equations for  $\hat{\mathbf{h}}_i$  and  $\hat{y}_i$ .

$$\hat{\mathbf{h}}_i = \tilde{\mathbf{h}}_i - \frac{(\mathbf{R}_{h_i h_i} \hat{\mathbf{x}} - \mathbf{R}_{h_i y_i}) e_i}{\hat{\mathbf{z}}^T \mathbf{R}_i \hat{\mathbf{z}}} \quad (40)$$

$$\hat{y}_i = \tilde{y}_i - \frac{(\mathbf{R}_{h_i y_i}^T \hat{\mathbf{x}} - R_{y_i y_i}) e_i}{\hat{\mathbf{z}}^T \mathbf{R}_i \hat{\mathbf{z}}} \quad (41)$$

Equations 40 and 41 yield the appropriate estimates for the measurements. Therefore we can substitute the optimal solution found in 39 into the original constrained objective function.

$$J(\hat{\mathbf{x}}) = \frac{1}{2} \sum_{i=1}^M \left( \tilde{\mathbf{d}}_i - \left( \tilde{\mathbf{d}}_i - \frac{\mathbf{R}_i \hat{\mathbf{z}}^T e_i}{\hat{\mathbf{z}}^T \mathbf{R}_i \hat{\mathbf{z}}} \right) \right)^T \mathbf{R}_i^{-1} \left( \tilde{\mathbf{d}}_i - \left( \tilde{\mathbf{d}}_i - \frac{\mathbf{R}_i \hat{\mathbf{z}}^T e_i}{\hat{\mathbf{z}}^T \mathbf{R}_i \hat{\mathbf{z}}} \right) \right) \quad (42)$$

Simplifying and performing the multiplication, we obtain the following form of the objective function

$$J(\hat{\mathbf{x}}) = \frac{1}{2} \sum_{i=1}^M \frac{e_i^T (\hat{\mathbf{z}} \mathbf{R}_i \hat{\mathbf{z}}^T) e_i}{(\hat{\mathbf{z}}^T \mathbf{R}_i \hat{\mathbf{z}})^2} = \frac{1}{2} \sum_{i=1}^M \frac{e_i^2}{\hat{\mathbf{z}}^T \mathbf{R}_i \hat{\mathbf{z}}} = \frac{1}{2} \sum_{i=1}^M \frac{(\tilde{\mathbf{h}}_i^T \hat{\mathbf{x}} - \tilde{y}_i)^2}{\hat{\mathbf{x}}^T \mathbf{R}_{h_i h_i} \hat{\mathbf{x}} - 2\mathbf{R}_{h_i y_i} \hat{\mathbf{x}} + R_{y_i y_i}} \quad (43)$$

Now we desire the optimal solution of  $\hat{\mathbf{x}}$  from the above equation, therefore we take its derivative and set it equal to zero as follows:

$$\frac{dJ(\hat{\mathbf{x}})}{d\hat{\mathbf{x}}} = \sum_{i=1}^M \frac{(\tilde{\mathbf{h}}_i \hat{\mathbf{x}}^{(j+1)} - \tilde{y}_i) \tilde{\mathbf{h}}_i}{\hat{\mathbf{x}}^T \mathbf{R}_{h_i h_i} \hat{\mathbf{x}} - 2\mathbf{R}_{h_i y_i} \hat{\mathbf{x}} + R_{y_i y_i}} - \frac{e_i (\hat{\mathbf{x}}^{(j)})^2 (\mathbf{R}_{h_i h_i} \hat{\mathbf{x}}^{(j+1)} - \mathbf{R}_{h_i y_i})}{(\hat{\mathbf{x}}^T \mathbf{R}_{h_i h_i} \hat{\mathbf{x}} - 2\mathbf{R}_{h_i y_i} \hat{\mathbf{x}} + R_{y_i y_i})^2} \quad (44)$$

The denominator is denoted as,  $\gamma_i = \hat{\mathbf{x}}^T \mathbf{R}_{h_i h_i} \hat{\mathbf{x}} - 2\mathbf{R}_{h_i y_i} \hat{\mathbf{x}} + R_{y_i y_i}$  and  $j$  denotes the time step. We must then solve for the estimate at the next time step,  $(j+1)$  as follows:

$$\hat{\mathbf{x}}^{(j+1)} = \left[ \sum_{i=1}^M \frac{\tilde{\mathbf{h}}_i \tilde{\mathbf{h}}_i^T}{\gamma_i (\hat{\mathbf{x}}^{(j)})} - \frac{e_i^2 (\hat{\mathbf{x}}^{(j)}) \mathbf{R}_{h_i h_i}}{\gamma_i^2 (\hat{\mathbf{x}}^{(j)})} \right]^{-1} \left[ \sum_{i=1}^M \frac{\tilde{y}_i \tilde{\mathbf{h}}_i}{\gamma_i (\hat{\mathbf{x}}^{(j)})} - \frac{e_i^2 (\hat{\mathbf{x}}^{(j)}) \mathbf{R}_{h_i y_i}}{\gamma_i^2 (\hat{\mathbf{x}}^{(j)})} \right] \quad (45)$$

No closed-form solution is available for the updated estimate, and an initial estimate must be given. This initial estimate can be given by any reasonable estimate such as the standard Least Squares solution or a variation of the Total Least Squares with more restrictive assumptions on the covariance matrix. In the next section we will implement both the Least Squares and Total Least Squares solutions previously defined for a simple line fitting problem.

### 3.8 LS and TLS Comparison

First recall the differences between the two solutions, the Least Squares minimizes the vertical distance whereas the Total Least Squares for the simulation which will be presented here where the variance in  $x$  and  $y$  are equal, minimizes the perpendicular distance. Equations 46 and 47 represent these errors, respectively, which will be computed for comparison.

$$d_v = y_i - (\hat{m}x_i + \hat{b}) \quad (46)$$

$$d_p = \frac{|\hat{m}x_i - y_i + \hat{b}|}{\sqrt{\hat{m}^2 + 1}} \quad (47)$$

Equation 47 can be obtained by projecting the vector from the point of interest,  $i$ , to the line, onto a perpendicular line. This is shown in the next steps and is also given by WolframMathWorld [20]. Suppose the line of interest is defined by the Equation 48

$$ax + by + c = 0 \quad (48)$$

Which can be manipulated into the standard slope-intercept form of a line. The point of interest, in which the perpendicular distance is to be minimized is denoted as,  $(x_0, y_0)$ . Rewriting in slope-intercept form and solving for  $x$  and  $y$  in terms of  $x$  gives the vector equations in 49.

$$\begin{bmatrix} x \\ -\frac{a}{b}x - \frac{c}{b} \end{bmatrix} = \begin{bmatrix} 0 \\ -\frac{c}{b} \end{bmatrix} - \frac{1}{b} \begin{bmatrix} -b \\ a \end{bmatrix} \quad (49)$$

These vector equations allow us to determine a vector which is perpendicular to the line,  $\mathbf{v}$  which is given by  $[a, b]^T$ . Finally we define the last vector,  $\mathbf{r}$ , which is a vector from the point,  $(x_0, y_0)$ , to a point on the line,  $(x, y)$ .

$$\mathbf{r} = \begin{bmatrix} x - x_0 \\ y - y_0 \end{bmatrix} \quad (50)$$

The projection of  $\mathbf{r}$  onto  $\mathbf{v}$  is a standard equation in linear algebra given by Equation 51

$$d_p = \frac{|\mathbf{v} \cdot \mathbf{r}|}{\|\mathbf{v}\|} \quad (51)$$

Figure 24 depicts the mathematical equations in a graphical form.

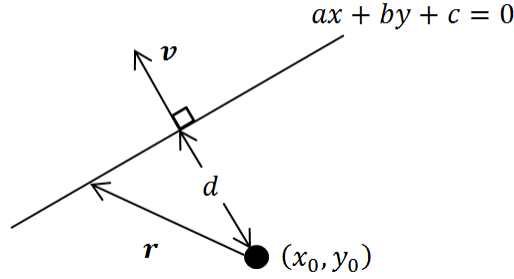


Fig. 24: Perpendicular Distance Point to Line [20]

Expanding the dot product in Equation 51 results in Equation 47. Now we present an example in which the Least Squares and Total Least Squares solutions are used. A line is defined by its slope,  $m = 1.243$  and intercept,  $b = -3.246$  with a range of  $x$  values of  $(-10, 10)$  and a step size of 0.1. All of the covariance matrices,  $\mathbf{R}$ , for each measurement are assumed to be equivalent and are defined as:

$$\mathbf{R}_{ls(M \times M)} = \sigma_{yy} \mathbf{I}_{(M \times M)} = 0.25 \times \mathbf{I}_{(M \times M)} \quad (52)$$

$$\mathbf{R}_{tls_i} = \begin{bmatrix} \sigma_{xx}^2 & 0 & \sigma_{xy}^2 \\ 0 & 0 & 0 \\ \sigma_{yx}^2 & 0 & \sigma_{yy}^2 \end{bmatrix} = \begin{bmatrix} 0.25 & 0 & 0.01 \\ 0 & 0 & 0 \\ 0.01 & 0 & 0.25 \end{bmatrix} \quad (53)$$



Where  $\mathbf{I}_{(M \times M)}$  is the  $M \times M$  identity matrix and  $M$  denotes the number of measurements. The true  $y$  values are generated using the true slope, intercept, and  $x$  values. In order to develop the measurements, white additive Gaussian noise is generated with the above specified covariances for each measurement,  $i = 1, 2, \dots, M$ . Figure 25 shows the true fit of the line in blue, using the given slope and intercept, the generated measurements in black, the Least squares solution in red, and the Total Least Squares solution in green. The plot has been zoomed to the region shown, for a better view of the comparison in values.

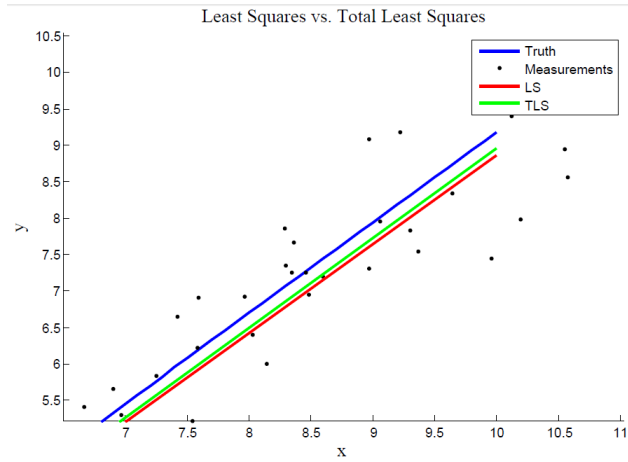


Fig. 25: Least Squares and Total Least Squares Comparison (zoom)

Table II shows the parameter estimates for each solution including the truth and in addition the root mean square error is computed using Equations 46 and 47 as the errors. As shown in the table the error in the estimates for the slope and intercept are smaller in the Total Least Squares solution. Also the RMSE in the vertical distance is better with the Least Squares while the RMSE in the perpendicular distance is minimal in the Total Least Squares, as expected.

TABLE II: Parameter Values and RMSE

Solution	$m$	$b$	Error (m)	Error (b)	RMS (Vertical)	RMS (Perpendicular)
True	1.243	-3.246	-	-	-	-
LS	1.2097	-3.2683	0.0333	0.0223	1.2014	0.7805
TLS	1.2317	-3.2681	0.0113	0.0221	1.2083	0.7801

### 3.9 Ellipse Fitting

Here we present the derivation for the Least Squares fitting of an ellipse to a set of data. This is presented by Fitzgibbon and Fischer [17] and is further developed by Halir and Flusser [18]. This method assumes that all of the data passed into the algorithm is associated with the ellipse and that only one ellipse is present in the data. Should there exist only a segment of an ellipse defined by the data, the algorithm will compute the parameters assuming an entire ellipse is to be identified. We begin with the basic equation for the conic section:

$$F(x, y) = ax^2 + bxy + cy^2 + dx + ey + f = 0 \quad (54)$$

In order to restrict the estimates of the coefficients to define an ellipse, the following constraint must be taken into account:

$$b^2 - 4ac < 0 \quad (55)$$

In terms of the least squares sense we define the vectors:

$$\mathbf{H} = [x^2, xy, y^2, x, y, 1] \quad (56)$$

$$\mathbf{x} = [a, b, c, d, e, f]^T \quad (57)$$

With these vectors we can rewrite the original equation to,  $F(\mathbf{H}, \mathbf{x}) = \mathbf{H}\mathbf{x}$ . Now we want to minimize the sum of the squared distances between the measurements and the conic section, or in this case the ellipse. Assuming there are  $M$  measurements, the constrained objective function to minimize the distance, by determining the coefficients of the conic section,  $\mathbf{x}$ , is given by Equation 58:

$$\arg \min_{\mathbf{x}} \quad J = \sum_{i=1}^M F(\mathbf{h}_i)^2 \quad (58)$$

$$\text{Subject to} \quad 4ac - b^2 = 1 \quad (59)$$

Where  $\mathbf{h}_i$  is the  $i$ th row of the  $\mathbf{H}$  matrix. The constraint has been reformulated into an equality constraint by introducing an arbitrary surplus variable, which is equated to one. This is possible because, for the set of coefficients which define a specific ellipse,  $\mathbf{x}$ , they can be scaled by a factor,  $\alpha$ , which forces the equality constraint,  $\alpha\mathbf{x} = 1$ . We prefer equality constraints since now the constrained optimization problem can be solved using the Lagrange multiplier method. In

order to solve this problem we must first define a series of matrices,  $\mathbf{H}$  which is a  $M \times 6$  matrix of measurements, and a  $6 \times 6$  constraint matrix,  $\mathbf{C}$ .

$$\mathbf{H} = \begin{bmatrix} x_1^2 & x_1 y_1 & y_1^2 & x_1 & y_1 & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x_i^2 & x_i y_i & y_i^2 & x_i & y_i & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x_M^2 & x_M y_M & y_M^2 & x_M & y_M & 1 \end{bmatrix} \quad \mathbf{C} = \begin{bmatrix} 0 & 0 & 2 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & 0 & 0 \\ 2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \quad (60)$$

The objective function is rewritten using the above matrices:

$$\arg \min_{\mathbf{x}} \quad J = \sum_{i=1}^M \|\mathbf{H}\mathbf{x}\|^2 \quad (61)$$

$$\text{Subject to} \quad \mathbf{x}^T \mathbf{C} \mathbf{x} = 1 \quad (62)$$

We multiply the constraint by the Lagrange multiplier,  $\lambda$  and append it to the objective function in order to obtain the unconstrained optimization function.

$$J = \mathbf{x}^T \mathbf{H}^T \mathbf{H} \mathbf{x} + \lambda (1 - \mathbf{x}^T \mathbf{C} \mathbf{x}) \quad (63)$$

The next step is to differentiate the unconstrained objective function with respect to the coefficient vector,  $\mathbf{x}^T$ :

$$\frac{dJ}{d\mathbf{x}^T} = \mathbf{H}^T \mathbf{H} \mathbf{x} - \lambda \mathbf{C} \mathbf{x} = 0 \quad (64)$$

Solving for the Lagrange multiplier,  $\lambda$  by recalling the constraint equation,  $\mathbf{x}^T \mathbf{C} \mathbf{x} = 1$  and left multiplying the previous equation by  $\mathbf{x}^T$  we get the solution for the Lagrange multiplier:

$$\lambda = \mathbf{x}^T \mathbf{S} \mathbf{x} \quad (65)$$

Where the matrix,  $\mathbf{S} = \mathbf{H}^T \mathbf{H}$ , is termed the scatter matrix. This is a simple eigenvalue problem in which we desire the minimal positive eigenvalue or Lagrange multiplier,  $\lambda$ . There exists up to six solutions to Equation 65. The eigenvector which corresponds to the minimal positive eigenvalue is the solution to the minimization problem and is the best estimate of the coefficient vector,  $\mathbf{x}$ . Halir and Flusser improve this algorithm by solving a set of equations which are obtained by partitioning the matrices,  $\mathbf{S}$ ,  $\mathbf{H}$ , and  $\mathbf{C}$  as well as the coefficient vector,  $\mathbf{x}$  into their linear and quadratic terms. They do this because the constraint matrix,  $\mathbf{C}$ , is singular and the scatter matrix,

$\mathbf{S}$ , can also be singular, though very rarely, and only if all of the measurements lie precisely on the ellipse. We begin with the partitioning of the  $\mathbf{H}$  matrix into its linear and quadratic terms as such:

$$\mathbf{H}_1 = \begin{bmatrix} x_1^2 & x_1 y_1 & y_1^2 \\ \vdots & \vdots & \vdots \\ x_i^2 & x_i y_i & y_i^2 \\ \vdots & \vdots & \vdots \\ x_M^2 & x_M y_M & y_M^2 \end{bmatrix} \quad \mathbf{H}_2 = \begin{bmatrix} x_1 & y_1 & 1 \\ \vdots & \vdots & \vdots \\ x_i & y_i & 1 \\ \vdots & \vdots & \vdots \\ x_M & y_M & 1 \end{bmatrix} \quad (66)$$

The remainder of the matrices are partitioned accordingly:

$$\mathbf{S} = \left[ \begin{array}{c|c} \mathbf{S}_1 = \mathbf{H}_1^T \mathbf{H}_1 & \mathbf{S}_2 = \mathbf{H}_1^T \mathbf{H}_2 \\ \hline \mathbf{S}_2^T = \mathbf{H}_2^T \mathbf{H}_1 & \mathbf{S}_3 = \mathbf{H}_2^T \mathbf{H}_2 \end{array} \right] \quad \mathbf{C} = \left[ \begin{array}{c|c} \mathbf{C}_1 = \begin{bmatrix} 0 & 0 & 2 \\ 0 & -1 & 0 \\ 2 & 0 & 0 \end{bmatrix} & \mathbf{0} \\ \hline \mathbf{0} & \mathbf{0} \end{array} \right] \quad (67)$$

$$\mathbf{x} = \left[ \begin{array}{c} \mathbf{x}_1 = \begin{bmatrix} a \\ b \\ c \end{bmatrix} \\ \hline \mathbf{x}_2 = \begin{bmatrix} d \\ e \\ f \end{bmatrix} \end{array} \right] \quad (68)$$

Solving Equation 64 using the partitioned matrices results in the set of two equations:

$$\mathbf{S}_1 \mathbf{x}_1 + \mathbf{S}_2 \mathbf{x}_2 = \lambda \mathbf{C}_1 \mathbf{x}_1 \quad (69)$$

$$\mathbf{S}_2^T \mathbf{x}_1 + \mathbf{S}_3 \mathbf{x}_2 = \mathbf{0} \quad (70)$$

We recall the definition of the linear measurements matrix,  $\mathbf{S}_3 = \mathbf{H}_2^T \mathbf{H}_2$ . This matrix is singular only when all the measurements precisely fit a line, again this scenario for an ellipse is rare

unless only points on the long straight segments exist in the measurement matrix. Therefore, under the assumption that this is not the case, we can solve for  $\mathbf{x}_2$  by taking the inverse of  $\mathbf{S}_3$ :

$$\mathbf{x}_2 = -\mathbf{S}_3^{-1}\mathbf{S}_2^T\mathbf{x}_1 \quad (71)$$

Substituting Equation 71 into 69 and since  $\mathbf{C}_1$  is non-singular we get:

$$\mathbf{C}_1^{-1}(\mathbf{S}_1 - \mathbf{S}_2\mathbf{S}_3^{-1}\mathbf{S}_2^T)\mathbf{x}_1 = \lambda\mathbf{x}_1 \quad (72)$$

This problem has been reduced from six-dimensions to three, since  $\mathbf{x}_2$  is now a function of  $\mathbf{x}_1$ . The following set of equations is easily solved by obtaining the eigenvector and eigenvalue which pertains to the smallest positive number.

$$\mathbf{M}\mathbf{x}_1 = \lambda\mathbf{x}_1 \quad (73)$$

$$\mathbf{x}_1^T\mathbf{C}_1\mathbf{x}_1 = 1 \quad (74)$$

$$\mathbf{x}_2 = -\mathbf{S}_3^{-1}\mathbf{S}_2^T\mathbf{x}_1 \quad (75)$$

$$\mathbf{x} = \begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \end{bmatrix} \quad (76)$$

Where  $\mathbf{M} = \mathbf{C}_1^{-1}(\mathbf{S}_1 - \mathbf{S}_2\mathbf{S}_3^{-1}\mathbf{S}_2^T)$ . This gives the coefficients of a conic section constrained to an ellipse, however in more practical scenarios we desire the parameters in terms of the center,  $(x_0, y_0)$ , major axis  $a'$ , minor axis  $b'$  and rotation angle,  $\phi$ . The equation of an ellipse including rotation is expressed as:

$$\frac{[(x - x_0) \cos \phi + (y - y_0) \sin \phi]^2}{a'^2} + \frac{[(x - x_0) \sin \phi + (y - y_0) \cos \phi]^2}{b'^2} = 1 \quad (77)$$

In order to transform the coefficients in  $\mathbf{x}$  into the desired parameters the following equations are used from WolframMathWorld [19]:

$$x_0 = \frac{cd - be}{b^2 - ac} \quad (78)$$

$$y_0 = \frac{ae - bd}{b^2 - ac} \quad (79)$$

$$a' = \frac{2(ae^2 + cd^2 + fb^2 - 2bde - acf)}{(b^2 - ac) \left[ \sqrt{(a - c)^2 + 4b^2} - (a + c) \right]} \quad (80)$$

$$b' = \frac{2(ae^2 + cd^2 + fb^2 - 2bde - acf)}{(b^2 - ac) \left[ -\sqrt{(a - c)^2 + 4b^2} - (a + c) \right]} \quad (81)$$

$$\phi = \begin{cases} 0 & \text{for } b = 0 \text{ and } a < c. \\ \frac{1}{2}\pi & \text{for } b = 0 \text{ and } a > c. \\ \frac{1}{2} \cot^{-1} \left( \frac{a-c}{2b} \right) & \text{for } b \neq 0 \text{ and } a < c. \\ \frac{\pi}{2} + \frac{1}{2} \cot^{-1} \left( \frac{a-c}{2b} \right) & \text{for } b \neq 0 \text{ and } a > c. \end{cases} \quad (82)$$

### 3.10 LS Ellipse Example

In this section we explore the capabilities of the least squares ellipse fitting algorithm defined by the equations in the previous section. Since this algorithm is not specific to the Hough Transform, it does not require the input to be a binary image, instead it simply requires the a set of coordinates which are to be considered part of the ellipse. The only assumption is that all of the data points passed into the algorithm must pertain to the ellipse, as all of these points will be considered when determining the coefficients. Since the algorithm which will be developed in the later sections, relies on the input being a binary image, due to the use of the Hough Transform, we will keep the same convention and use an image of an ellipse as our example data. Figure 26 displays the generated ellipse (via the paint program) and the results of the ellipse Least Squares algorithm. Keeping in mind that a generated ellipse using an image is a series of square pixels, the algorithm fits an ellipse to the set of extracted data points.

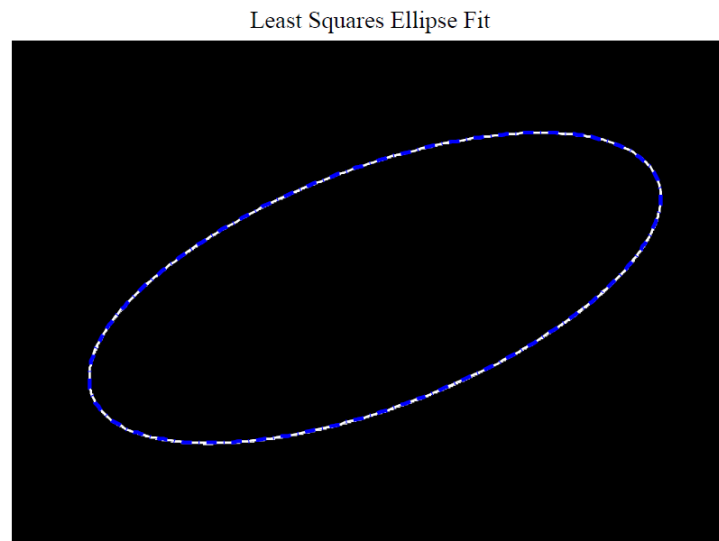


Fig. 26: Least Squares Ellipse Fitting Results

### 3.11 Uncertainty Analysis

In this section we will present the derivation for the Cramer Rao lower bounds which provides a measure of uncertainty in the coefficients of the linear, curve, and ellipse fits. These derivations are based on fully populated covariance matrices. The bounds are estimated by the inverse of the Fisher Information matrix. We begin by deriving the bounds for the line fit and then for the blending functions, third order polynomials in  $x$ . We conclude the derivations with the bounds for an ellipse defined by all five parameters. Each set of derivations are accompanied by a Monte Carlo simulation to verify the convergence properties of the solutions.

### 3.12 Straight Line

For the straight line we have a total of three unknown parameters, the slope,  $m$ , the intercept,  $b$ , and the true value of  $x$ ,  $x_t$ . The measurement functions are given as:

$$y = mx_t + b + \nu_y \quad (83)$$

$$x = x_t + \nu_x \quad (84)$$

The noise variables,  $\nu_y$  and  $\nu_x$ , are assumed to be Gaussian white noise and correlated, giving the fully populated covariance matrix,  $\mathbf{Q}$ .

$$\mathbf{Q} = \begin{bmatrix} \sigma_{yy}^2 & \sigma_{xy}^2 \\ \sigma_{yx}^2 & \sigma_{xx}^2 \end{bmatrix}$$

We begin by determining the estimate of the true value of  $x$ ,  $\hat{x}_t$ . This is done using the solution from [16] as previously presented. The necessary covariances are defined as follows:

$$\mathbf{R}_{h_i h_i} = \begin{bmatrix} \sigma_{x_i x_i}^2 & 0 \\ 0 & 0 \end{bmatrix} \quad \mathbf{R}_{h_i y_i} = \begin{bmatrix} \sigma_{x_i y_i}^2 & 0 \end{bmatrix} \quad R_{y_i y_i} = \sigma_{y_i y_i}^2 \quad (85)$$

Therefore the complete covariance matrix,  $\mathbf{R}$ , for the Total Least Squares algorithm is the augmented matrix of the above covariances:

$$\mathbf{R}_i = \begin{bmatrix} \mathbf{R}_{h_i h_i} & \mathbf{R}_{h_i y_i}^T \\ \mathbf{R}_{h_i y_i} & R_{y_i y_i} \end{bmatrix} = \begin{bmatrix} \sigma_{x_i x_i}^2 & 0 & \sigma_{x_i y_i}^2 \\ 0 & 0 & 0 \\ \sigma_{x_i y_i}^2 & 0 & \sigma_{y_i y_i}^2 \end{bmatrix} \quad (86)$$

We now recall the solutions for the estimates of  $\hat{\mathbf{h}}_i$  and  $\hat{y}_i$ , Equations 40 and 41 respectively. Next the equations are expanded by implementing the matrix-vector multiplications and noting the



definitions, the covariance matrix,  $\mathbf{R}$ , the coefficient vector,  $\hat{\mathbf{x}} = [\hat{m}, \hat{b}]^T$ , and the measurements and their respective estimated true values,  $\tilde{\mathbf{h}}_i = [x_i, 1]$ ,  $\hat{\mathbf{h}}_i = [\hat{x}_{t_i}, 1]$ ,  $\tilde{y} = y_i$ , and  $\hat{y}_i = \hat{y}_{t_i}$ . Thus for a single measurement,  $i$ , we can calculate the estimated values of  $x_{t_i}$  and  $y_{t_i}$  as given by Equations 87 and 88 respectively.

$$\hat{x}_{t_i} = x_i - \frac{(\hat{m}\sigma_{x_i x_i}^2 - \sigma_{x_i y_i}^2) e_i}{\hat{m}^2 \sigma_{x_i x_i}^2 - 2\hat{m}\sigma_{x_i y_i}^2 + \sigma_{y_i y_i}^2} \quad (87)$$

$$\hat{y}_{t_i} = y_i - \frac{(\hat{m}\sigma_{x_i y_i}^2 - \sigma_{y_i y_i}^2) e_i}{\hat{m}^2 \sigma_{x_i x_i}^2 - 2\hat{m}\sigma_{x_i y_i}^2 + \sigma_{y_i y_i}^2} \quad (88)$$

Where we recall that,  $e_i$  is the error in the measurement with the appropriate estimates,  $e_i = \hat{m}x_i + \hat{b} - y_i$ . Note that in Equations 87 and 88, the estimate for the slope and intercept must be known in order to calculate  $\hat{x}_{t_i}$  and  $\hat{y}_{t_i}$ . In our scenario these initial estimates are given by the basic Least Squares solution. Since in the Hough transform, should the value of  $\theta$  be exactly zero (due to the quantization), if we convert the parameters into the necessary slope and intercept values, infinite values will be obtained. This can be seen in Equations 89 and 90

$$\hat{m} = -\frac{\cos \theta}{\sin \theta} \quad (89)$$

$$\hat{b} = \frac{\rho}{\sin \theta} \quad (90)$$

With the initial estimates for the unknown parameters, slope and intercept, one can obtain the estimated true value,  $\hat{x}_{t_i}$ , using Equation 87. In addition to the estimated true value of  $x$ , the line's coefficients are updated using the Total Least Squares solution. The next step is to determine the uncertainty in these estimates. The probability density function for a single measurement,  $(x_i, y_i)$ , given the estimates  $[\hat{m}, \hat{b}, \hat{x}_{t_i}]$  is given by:

$$p(x_i, y_i | \hat{m}, \hat{b}, \hat{x}_{t_i}) = \frac{1}{\sqrt{2\pi|\mathbf{Q}_i|}} e^{-\frac{1}{2} \left( \begin{bmatrix} y_i \\ x_i \end{bmatrix} - \begin{bmatrix} \hat{m}\hat{x}_{t_i} + \hat{b} \\ \hat{x}_{t_i} \end{bmatrix} \right)^T \mathbf{Q}_i^{-1} \left( \begin{bmatrix} y_i \\ x_i \end{bmatrix} - \begin{bmatrix} \hat{m}\hat{x}_{t_i} + \hat{b} \\ \hat{x}_{t_i} \end{bmatrix} \right)} \quad (91)$$

The inverse of the covariance matrix,  $\mathbf{Q}_i$  is easily defined for a  $2 \times 2$  matrix:

$$\mathbf{Q}_i^{-1} = \frac{1}{\sigma_{x_i x_i}^2 \sigma_{y_i y_i}^2 - \sigma_{x_i y_i}^4} \begin{bmatrix} \sigma_{y_i y_i}^2 & -\sigma_{x_i y_i}^2 \\ -\sigma_{x_i y_i}^2 & \sigma_{x_i x_i}^2 \end{bmatrix} \quad (92)$$

We expand the exponential term in Equation 91 and write it as a product of two variables,  $\alpha_i$  and  $K_i$  which are defined in Equations 93 and 94, respectively.

$$\alpha_i = -\frac{1}{2(\sigma_{x_i x_i}^2 \sigma_{y_i y_i}^2 - \sigma_{x_i y_i}^2)} \quad (93)$$

$$K_i = \left( \sigma_{x_i x_i}^2 \left( y_i - (\hat{m} \hat{x}_{t_i} + \hat{b}) \right)^2 - 2\sigma_{x_i y_i}^2 \left( y_i - (\hat{m} \hat{x}_{t_i} + \hat{b}) \right) (x_i - \hat{x}_{t_i}) + \sigma_{y_i y_i}^2 (x_i - \hat{x}_{t_i})^2 \right) \quad (94)$$

With  $\alpha_i$  and  $K_i$  defined for each measurement we can then rewrite the likelihood function in a more contracted form.

$$p(x_i, y_i | \hat{m}, \hat{b}, \hat{x}_{t_i}) = \frac{1}{\sqrt{2\pi|\mathbf{Q}_i|}} e^{\alpha_i K_i} \quad (95)$$

Each measurement,  $i$ , is independent of one another. Therefore if we assume there are a total of  $M$  measurements, the probability density function for the matrix of measurements,  $[\mathbf{x}, \mathbf{y}]$ , given the parameter estimates,  $[\hat{m}, \hat{b}, \hat{\mathbf{x}}_t]$ , where  $\hat{\mathbf{x}}_t$  is now a vector of estimated  $x$  values, is given by the product of each measurement's probability density function:

$$p(\mathbf{x}, \mathbf{y} | \hat{m}, \hat{b}, \hat{\mathbf{x}}_t) = \prod_{i=1}^M \frac{1}{\sqrt{2\pi(\sigma_{x_i x_i}^2 \sigma_{y_i y_i}^2 - \sigma_{x_i y_i}^2)}} e^{\alpha_i K_i} \quad (96)$$

The Fisher Information matrix is defined as the negative expected value of the Hessian of the log-likelihood function with respect to the estimated parameters. We define the log-likelihood function as,  $f = \ln p(\mathbf{x}, \mathbf{y} | \hat{m}, \hat{b}, \hat{\mathbf{x}}_t)$ , therefore the Fisher Information matrix is defined as:

$$F = -E \begin{bmatrix} \frac{d^2 f}{d\hat{m}d\hat{m}} & \frac{d^2 f}{d\hat{m}d\hat{b}} & \frac{d^2 f}{d\hat{m}d\hat{x}_t} \\ \frac{d^2 f}{d\hat{b}d\hat{m}} & \frac{d^2 f}{d\hat{b}d\hat{b}} & \frac{d^2 f}{d\hat{b}d\hat{x}_t} \\ \frac{d^2 f}{d\hat{x}_t d\hat{m}} & \frac{d^2 f}{d\hat{x}_t d\hat{b}} & \frac{d^2 f}{d\hat{x}_t d\hat{x}_t} \end{bmatrix} \quad (97)$$

First the partial derivatives of  $f$  are taken with respect to the estimated parameters. Recall that the probability density function is a product of the individual measurement's and with the properties of the log the product becomes a summation over the  $M$  measurements. Thus the

partial derivatives are:

$$\frac{df}{d\hat{m}} = \sum_{i=1}^M \alpha_i \left[ -2\sigma_{x_i y_i}^2 (-x_i \hat{x}_{t_i} + \hat{x}_{t_i}^2) + \sigma_{x_i x_i}^2 (-2y_i \hat{x}_{t_i} + 2\hat{m} \hat{x}_{t_i}^2 + 2\hat{x}_{t_i} \hat{b}) \right] \quad (98)$$

$$\frac{df}{d\hat{b}} = \sum_{i=1}^M \alpha_i \left[ -2\sigma_{x_i y_i}^2 (-x_i + \hat{x}_{t_i}) + \sigma_{x_i x_i}^2 (-2y_i + 2\hat{m} \hat{x}_{t_i} + 2\hat{b}) \right] \quad (99)$$

$$\begin{aligned} \frac{df}{d\hat{x}_t} = \sum_{i=1}^M \alpha_i & \left[ \sigma_{y_i y_i}^2 (-2x_i + 2\hat{x}_{t_i}) - 2\sigma_{x_i y_i}^2 (-y_i - x_i \hat{m} + 2\hat{m} \hat{x}_{t_i} + \hat{b}) + \dots \right. \\ & \left. + \sigma_{x_i x_i}^2 (-2y_i \hat{m} + 2\hat{m}^2 \hat{x}_{t_i} + 2\hat{m} \hat{b}) \right] \end{aligned} \quad (100)$$

Then the second derivatives which compose the Fisher Information matrix can easily be determined:

$$F(1, 1) = \frac{d^2 f}{d\hat{m} d\hat{m}} = \sum_{i=1}^M \alpha_i (2\sigma_{x_i x_i}^2 \hat{x}_{t_i}^2) \quad (101)$$

$$F(1, 2) = F(2, 1) = \frac{d^2 f}{d\hat{m} d\hat{b}} = \sum_{i=1}^M \alpha_i (2\sigma_{x_i x_i}^2 \hat{x}_{t_i}) \quad (102)$$

$$F(1, 3) = F(3, 1) = \frac{d^2 f}{d\hat{m} d\hat{x}_t} = \sum_{i=1}^M \alpha_i \left( -2\sigma_{x_i y_i}^2 (-x_i + 2\hat{x}_{t_i}) + \sigma_{x_i x_i}^2 (-2y_i + 4\hat{m} \hat{x}_{t_i} + 2\hat{b}) \right) \quad (103)$$

$$F(2, 2) = \frac{d^2 f}{d\hat{b} d\hat{b}} = \sum_{i=1}^M \alpha_i (2\sigma_{x_i x_i}^2) \quad (104)$$

$$F(2, 3) = F(3, 2) = \frac{d^2 f}{d\hat{x}_t d\hat{b}} = \sum_{i=1}^M \alpha_i (-\sigma_{x_i y_i}^2 + 2\sigma_{x_i x_i}^2 \hat{m}) \quad (105)$$

$$F(3, 3) = \frac{d^2 f}{d\hat{x}_t d\hat{x}_t} = \sum_{i=1}^M \alpha_i (2\sigma_{y_i y_i}^2 - 4\sigma_{x_i y_i}^2 \hat{m} + 2\sigma_{x_i x_i}^2 \hat{m}^2) \quad (106)$$

These equations then give us an estimate of the uncertainty in the estimated parameters. The next step is to perform a Monte Carlo simulation to show the convergence characteristics of this estimate. We begin the simulation by choosing a slope, intercept, and range of  $x$  values. These will be the true simulation parameters and are specified as:

$$m = -0.4326 \quad b = -1.6656 \quad x = -3 : 0.1 : 3 \quad (107)$$

Furthermore the covariance matrices,  $\mathbf{Q}$ , for each of the measurements is again assumed to be equal and is given to be:

$$\mathbf{Q} = \begin{bmatrix} \sigma_{yy}^2 & \sigma_{xy}^2 \\ \sigma_{yx}^2 & \sigma_{xx}^2 \end{bmatrix} = \begin{bmatrix} 0.5 & 0.01 \\ 0.01 & 0.5 \end{bmatrix} \quad (108)$$

Since this simulation is to determine the convergence characteristics and not the capabilities of the Hough Transform, we will use the Least Squares solution from Equation 27 where the covariance matrix,  $\mathbf{R}$  is  $\sigma_{yy}^2 \mathbf{I}_{M \times M}$ , where  $M$  is the number of measurements, which in this simulation is 61. Since the  $x$  truth was already established the  $y$  truth can be calculated using the given values for the true slope and intercept. Gaussian white noise is then added to the truth, which was specified in  $\mathbf{Q}$  and finally the estimated values of  $x$  and  $y$  can be obtained via Equations 87 and 88 respectively. A single simulation's results are shown in Figure 27. Here the truth is shown in blue, the measurements (truth with added noise) are black, and the Total Least Squares line fit is given by the red line.

The estimated values of the slope and intercept from the Total Least Squares algorithm, for this single simulation run are:

$$\hat{m} = -0.3775 \quad \hat{b} = -1.5871 \quad (109)$$

We perform 10,000 simulations to determine the convergence characteristics of the Fisher Information matrix. The measure used for convergence is the determinant of the difference of the Monte Carlo covariance and the inverse of the Fisher Information matrix. The Monte Carlo covariance is calculated as a difference of the truth and the averaged estimates. We will denote this covariance as  $MC_{cov}$  and is calculated as:

$$MC_{cov} = \frac{1}{10,000} \sum_{i=1}^{10,000} \left( \begin{bmatrix} m \\ b \\ x_{t_1} \end{bmatrix} - \begin{bmatrix} \bar{m} \\ \bar{b} \\ \bar{x}_{t_1} \end{bmatrix} \right) \left( \begin{bmatrix} m \\ b \\ x_{t_1} \end{bmatrix} - \begin{bmatrix} \bar{m} \\ \bar{b} \\ \bar{x}_{t_1} \end{bmatrix} \right)^T \quad (110)$$

Approved for Public Release; Distribution Unlimited.

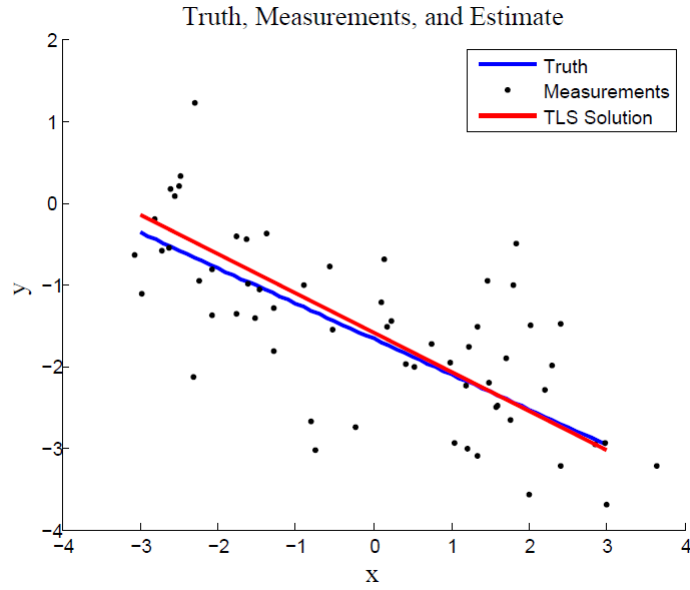


Fig. 27: Truth, Measurements, Estimate, TLS Line Fit

Where the estimated values of  $m$ ,  $b$ ,  $x_{t_1}$  are averaged after each simulation which are denoted by,  $\bar{\hat{m}}$ ,  $\bar{\hat{b}}$ , and  $\bar{\hat{x}_{t_1}}$  respectively. Then the convergence measure is given to be:

$$convergence = |MC_{cov} - F^{-1}| \quad (111)$$

The Fisher Information matrix is also averaged over each simulation. Figure 28 shows the value of this *convergence* measure after each simulation.

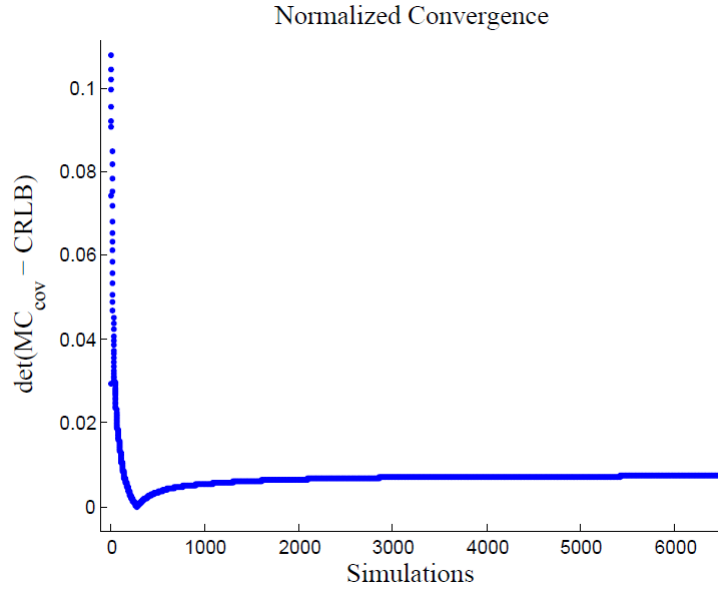


Fig. 28: Monte Carlo Simulation

Where in Figure 28 we have shown only a portion of the total number of simulations and have normalized the convergence value. The estimated bounds on the parameters, the inverse of the negative of the Fisher Information matrix after 10,000 simulations is given to be:

$$F^{-1} = \begin{bmatrix} 0.0011 & 0 & 0.0001 \\ 0 & 0.0056 & 0.0028 \\ 0.0001 & 0.0028 & 0.0055 \end{bmatrix}$$

Next we examine the estimates and their statistical properties. Each of the estimated values of  $m$  and  $b$  are plotted in Figure 29 along with the sigma ellipses.

Finally we select all combinations of  $\hat{m}$  and  $\hat{b}$  which fall within one sigma of the average values of the respective estimates and plot them, where the range is given as:

$$\hat{m} = -0.4381 \pm 0.0336 \quad (112)$$

$$\hat{b} = -1.6659 \pm 0.0719 \quad (113)$$

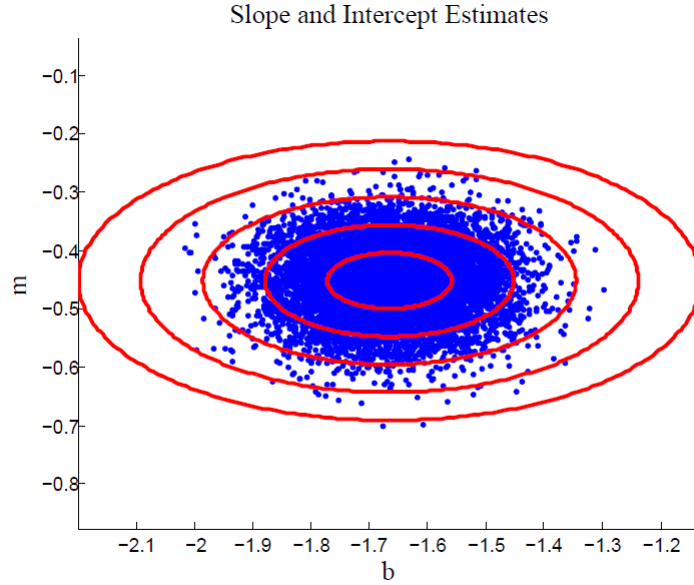


Fig. 29: Sigma Ellipses

The lines defined by all such coefficients are plotted, in blue, along with the true fit of the line, in red, in Figure 30 and we can see the uncertainty in the estimates. As we diverge from the relative midpoint of the data range the uncertainty grows. This is as to be expected since the variance in the  $y$  direction depends on the variance of the slope, intercept, and estimated  $x$  value.

To prove that the variance in the  $y$  direction is dependent on the variance of the slope, intercept, and estimated  $x$  value we can transform the Fisher Information matrix from the unknown parameters into a variance in terms of  $x$  and  $y$ . This can be done using the Jacobian transformation. In this transformation the Fisher Information matrix is left and right multiplied by the Jacobian of the measurement functions with respect to the estimated parameters. The Jacobian takes the form of:

$$A = \begin{bmatrix} \frac{dy}{dm} & \frac{dy}{db} & \frac{dy}{dx_{t_i}} \\ \frac{dx}{dm} & \frac{dx}{db} & \frac{dx}{dx_{t_i}} \end{bmatrix} = \begin{bmatrix} \hat{x}_{t_i} & 1 & \hat{m} \\ 0 & 0 & 1 \end{bmatrix} \quad (114)$$

Therefore we perform the matrix multiplication to understand the growing variance in the  $y$

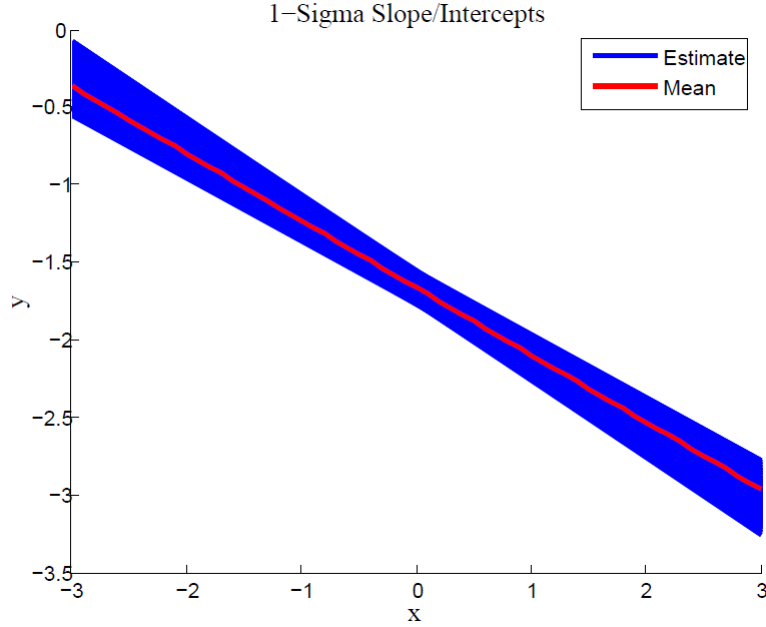


Fig. 30: One Sigma Slopes/Intercepts

direction:

$$\begin{bmatrix} \sigma_{y_i y_i}^2 & \sigma_{x_i y_i}^2 \\ \sigma_{x_i y_i}^2 & \sigma_{x_i x_i}^2 \end{bmatrix} = A_i F^{-1} A_i^T = \begin{bmatrix} \hat{x}_{t_i} & 1 & \hat{m} \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \sigma_{mm}^2 & \sigma_{mb}^2 & \sigma_{mx_t}^2 \\ \sigma_{bm}^2 & \sigma_{bb}^2 & \sigma_{bx_t}^2 \\ \sigma_{x_t m}^2 & \sigma_{x_t b}^2 & \sigma_{x_t x_t}^2 \end{bmatrix} \begin{bmatrix} \hat{x}_{t_i} & 0 \\ 1 & 0 \\ \hat{m} & 0 \end{bmatrix} \quad (115)$$

This results in the covariances taking the form of Equations 116 through 119

$$\sigma_{y_i y_i}^2 = \hat{x}_{t_i}^2 \sigma_{mm}^2 + 2\hat{x}_{t_i} \sigma_{mb}^2 + 2\hat{m} \hat{x}_{t_i} \sigma_{mx_t}^2 + \sigma_{bb}^2 + 2\hat{m} \sigma_{x_t b}^2 + \hat{m}^2 \sigma_{x_t x_t}^2 \quad (116)$$

$$\sigma_{x_i y_i}^2 = \hat{x}_{t_i} \sigma_{mx_t}^2 + \sigma_{bx_t}^2 + \hat{m} \sigma_{x_t x_t}^2 \quad (117)$$

$$\sigma_{y_i x_i}^2 = \sigma_{x_i y_i}^2 \quad (118)$$

$$\sigma_{x_i x_i}^2 = \sigma_{x_t x_t}^2 \quad (119)$$

From the above equations we can see that the variance in the y direction depends on the varying value of  $\hat{x}_{t_i}$ , so therefore as we diverge from  $\hat{x}_{t_i} = 0$  in either direction the variance in y grows.

### 3.13 Algebraic Fit Covariance

In this section we explain the derivation presented by Chernov and Lesort [21] concerning the covariance matrix of the weighted algebraic fit. The algebraic fit is more commonly known



as the minimization of the following summation:

$$\sum_{i=1}^n w_i [P(x_i, y_i; \Theta)]^2 \quad (120)$$

In this case  $x$  and  $y$  are always assumed to be the measurements and  $\Theta$  is the vector of parameters which will be defined for both scenarios. The weights,  $w$ , are assumed to be a function of  $x$ ,  $y$ , and the parameter vector,  $\Theta$ . The solution to Equation 120 must satisfy the equivalency to zero of its derivative with respect to the unknown parameter vector,  $\Theta$ . Using the chain rule Equation 121 is obtained.

$$\sum P_i^2 \nabla_{\Theta} w_i + 2 \sum w_i P_i \nabla_{\Theta} P_i = 0 \quad (121)$$

The first summation is discarded since we are only solving with respect to the leading order. Furthermore in our case we will assume that the weights,  $w$ , are all equal to one. Therefore the first term in Equation 121 is nullified regardless of the leading order assumption.

$$\sum w_i P_i \nabla_{\Theta} P_i = 0 \quad (122)$$

Since the true points must lie on the true curve, where true is denoted by,  $\bar{\mathbf{x}}$  and  $\bar{\Theta}$ , using the chain rule, Equation 123 can be viewed as an additional constraint on the system.

$$\langle \nabla_{\mathbf{x}} P(\bar{\mathbf{x}}_i; \bar{\Theta}), \delta \bar{\mathbf{x}}_i \rangle + \langle \nabla_{\Theta} P(\bar{\mathbf{x}}_i; \bar{\Theta}), \delta \bar{\Theta} \rangle = 0 \quad (123)$$

Where the angled brackets denote a vector product. We can then substitute Equation 123 into Equation 122 and solve for the variance of the parameters,  $\delta \Theta$ .

$$\sum w_i \nabla_{\Theta} P_i \nabla_{\Theta} P_i^T \delta \Theta + \sum w_i \nabla_{\mathbf{x}} P_i^T \delta \mathbf{x}_i \nabla_{\Theta} P_i \quad (124)$$

Solving for the variance term results in Equation 125

$$\delta \Theta = - \left[ \sum w_i \nabla_{\Theta} P_i \nabla_{\Theta} P_i^T \right]^{-1} \left[ \sum w_i \nabla_{\mathbf{x}} P_i^T \delta \mathbf{x}_i \nabla_{\Theta} P_i \right] \quad (125)$$

The covariance can be obtained via Equation 126.

$$C_{\Theta} = E [\delta \Theta \delta \Theta^T] \quad (126)$$

### 3.14 3rd Order Polynomial

The 3rd order polynomial is used as the blending function between two line segments. This polynomial takes the form of Equation 127.

$$y = Ax^3 + Bx^2 + Cx + D \quad (127)$$

We assume that there is Gaussian white noise in both  $x$  and  $y$ . Our desire is then to determine the uncertainty of the parameter estimates determined in the Least Squares approach. The parameter vector is defined as,  $\Theta = (A, B, C, D)$ . Using the derivations from Chernov and Lesort [21], in particular Equation 125, the desired variance terms can be obtained. In order to begin we must first define the gradient vectors with respect to both the parameters and the measurements.

$$\nabla_{\Theta} P_i = \begin{bmatrix} \frac{dP_i}{dA} \\ \frac{dP_i}{dB} \\ \frac{dP_i}{dC} \\ \frac{dP_i}{dD} \end{bmatrix} = \begin{bmatrix} x_i^3 \\ x_i^2 \\ x_i \\ 1 \end{bmatrix} \quad (128)$$

$$\nabla_{\mathbf{x}} P_i = \begin{bmatrix} \frac{dP_i}{dx} \\ \frac{dP_i}{dy} \end{bmatrix} = \begin{bmatrix} 3Ax_i^2 + 2Bx_i + C \\ -1 \end{bmatrix} \quad (129)$$

With the gradient vectors defined, the variance of the estimated parameters can easily be computed. Therefore we move onto a simulation to show the results of the derivations for the 3rd order polynomial in  $x$ . We begin by defining a set of parameters and a range on  $x$ .

$$A = 5 \quad B = 2 \quad C = -1 \quad D = 4 \quad x = -7 : 0.1 : 7 \quad (130)$$

In addition to the specified variables we must define the variance in the  $x$  and  $y$  terms.

$$\sigma_x^2 = 0.5 \quad \sigma_y^2 = 0.75 \quad (131)$$

The plotted results of the generated measurements, in black, along with the true polynomial, in blue, and the estimated fit, in red, using the built in MatLab function, *polyfit*, which uses a Least Squares algorithm to determine the coefficient estimates, are shown in Figure 31.

We then implement the derivations in Equation 125 and obtain the following covariance matrix

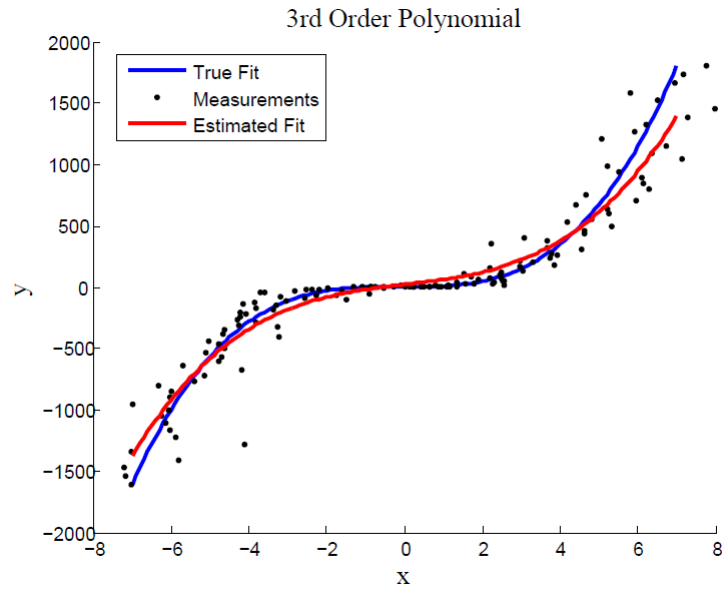


Fig. 31: 3rd Order Polynomial Simulation

for the estimated parameters:

$$C_{\Theta} = \begin{bmatrix} \sigma_{AA}^2 & \sigma_{AB}^2 & \sigma_{AC}^2 & \sigma_{AD}^2 \\ \sigma_{BA}^2 & \sigma_{BB}^2 & \sigma_{BC}^2 & \sigma_{BD}^2 \\ \sigma_{CA}^2 & \sigma_{CB}^2 & \sigma_{CC}^2 & \sigma_{DC}^2 \\ \sigma_{DA}^2 & \sigma_{DB}^2 & \sigma_{DC}^2 & \sigma_{DD}^2 \end{bmatrix} = \begin{bmatrix} 0.0040 & 0.0198 & -0.0562 & -0.1561 \\ 0.0198 & 0.0978 & -0.2778 & -0.7711 \\ -0.0562 & -0.2778 & 0.7895 & 2.1911 \\ -0.1561 & -0.7711 & 2.1911 & 6.0809 \end{bmatrix} \quad (132)$$

Figure 32 shows the 3-sigma bounded region for the estimated polynomial fit shown in red in Figure 31. The plotted bounds implemented here is the upper and lower 3-sigma bounds on each of the parameters simultaneously. As we defer from the origin the bounds expand rapidly. In order to plot the uncertainty we consider two simple cases. These cases are adding and subtracting the uncertainty to each of the four estimated parameters simultaneously, these are the two cases shown in Figure 32. The red center-line represents the mean of the estimated polynomials and the blue shaded region is the uncertainty region previously described.

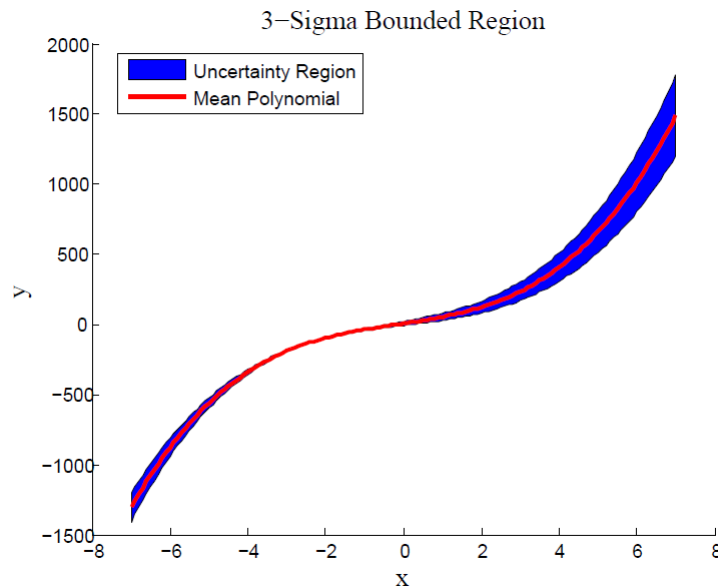


Fig. 32: 3rd Order Polynomial Uncertainty

As with the straight line case, we also present a Monte Carlo simulation. The addition of the noise along with the polynomial approximation and the estimation of the bounds is performed a total of 10,000 times. The convergence of the solution is determined again by a determinant of the difference of two matrices.

The first matrix is the averaged bounds computed by Equation 125 and the second matrix is the averaged Monte Carlo covariance which is calculated as in Equation 133, however the parameters now correspond to the 3rd order polynomial.

$$MC_{cov} = \frac{1}{10,000} \sum_{i=1}^{10,000} \left( \begin{bmatrix} A \\ B \\ C \\ D \end{bmatrix} - \begin{bmatrix} \bar{\bar{A}} \\ \bar{\bar{B}} \\ \bar{\bar{C}} \\ \bar{\bar{D}} \end{bmatrix} \right) \left( \begin{bmatrix} A \\ B \\ C \\ D \end{bmatrix} - \begin{bmatrix} \bar{\bar{A}} \\ \bar{\bar{B}} \\ \bar{\bar{C}} \\ \bar{\bar{D}} \end{bmatrix} \right)^T \quad (133)$$

Where the estimated coefficients are averaged after each simulation which are denoted by,  $\bar{\bar{A}}$ ,  $\bar{\bar{B}}$ ,  $\bar{\bar{C}}$ , and  $\bar{\bar{D}}$  respectively. Then the convergence measure is given to be:

$$convergence = |MC_{cov} - F^{-1}| \quad (134)$$

The convergence of the normalized determinant value is shown in Figure 33, for only a portion of the total number of simulations.

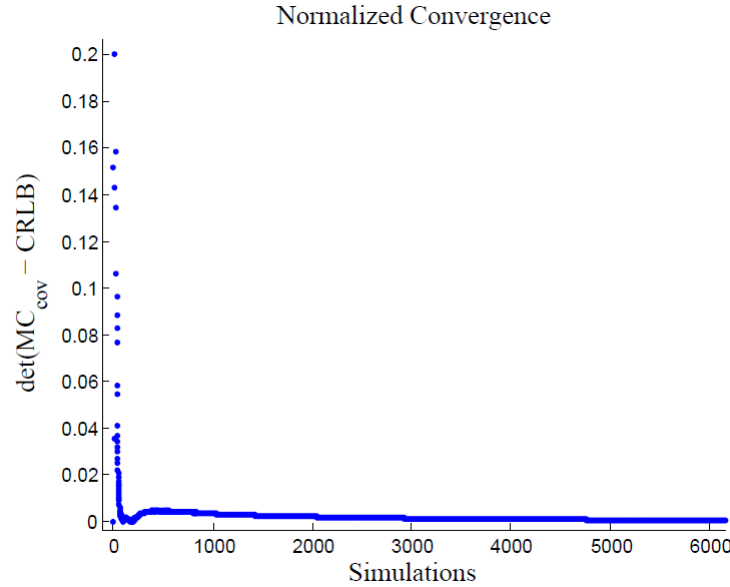


Fig. 33: 3rd Order Polynomial Uncertainty Convergence

In addition to Figure 32 we plot all polynomial fits with estimated parameters within 3-sigma of the mean. The mean values are given to be:

$$\bar{A} = 3.2252 \quad \bar{B} = 1.7460 \quad \bar{C} = 41.6646 \quad \bar{D} = 7.4202 \quad (135)$$

From the mean values we note that there is a large discrepancy in the value of  $C$ , since the true value was given to be  $-1$ . The 3-sigma polynomial fits are shown in Figure 34 in blue while the polynomial defined by the mean values of the parameters is shown in red.

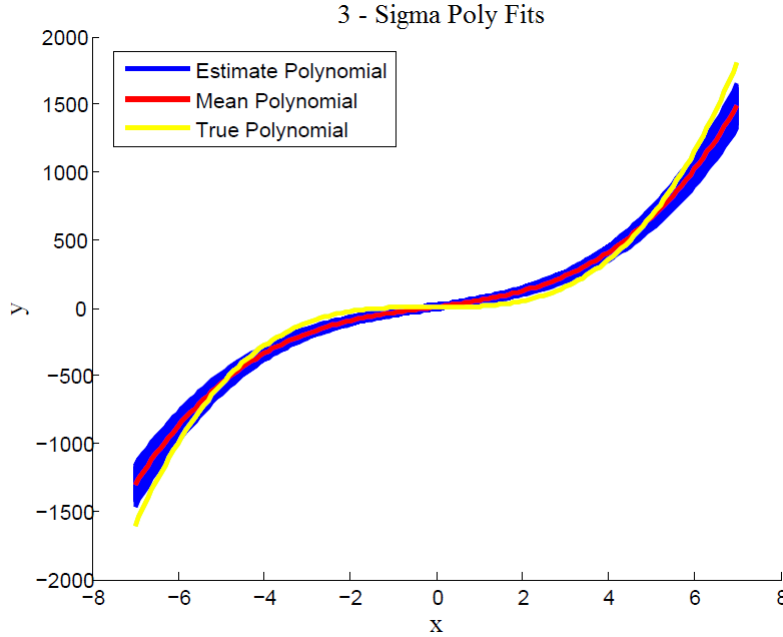


Fig. 34: 3-Sigma Polynomials

Also plotted in Figure 34 is the true curve, in yellow. From this we can see that even though there is a large discrepancy in the value of  $C$ , over the plotted range, there is only a slight difference in the polynomial fits.

### 3.15 Ellipse

Here we derive the uncertainty in the parameters of the ellipse equation presented in Equation 77 and is presented here again.

$$\frac{[(x - x_0) \cos \phi + (y - y_0) \sin \phi]^2}{a'^2} + \frac{[(x - x_0) \sin \phi + (y - y_0) \cos \phi]^2}{b'^2} = 1 \quad (136)$$

We now have five parameters,  $a'$ ,  $b'$ ,  $x_0$ ,  $y_0$  and  $\phi$ , in which to characterize the uncertainty. The parameter vector is defined as,  $\Theta = [a', b', x_0, y_0, \phi]$ . In order to determine the uncertainty we need to define the gradient vectors with respect to the parameters and the measurements. The

gradient vector with respect to the parameters is defined in Equation 137.

$$\nabla_{\Theta} P_i = \begin{bmatrix} \frac{dP_i}{da'} \\ \frac{dP_i}{db} \\ \frac{dP_i}{dx_0} \\ \frac{dP_i}{dy_0} \\ \frac{dP_i}{d\phi} \end{bmatrix} = \begin{bmatrix} -\frac{2((x-x_0)\cos\phi+(y-y_0)\sin\phi)^2}{a'^3} \\ -\frac{2((x-x_0)\sin\phi-(y-y_0)\cos\phi)^2}{b^3} \\ -\frac{2((x-x_0)\cos\phi+(y-y_0)\sin\phi)\cos\phi}{a'^2} - \frac{2((x-x_0)\sin\phi-(y-y_0)\cos\phi)\sin\phi}{b^2} \\ -\frac{2((x-x_0)\cos\phi+(y-y_0)\sin\phi)\sin\phi}{a'^2} - \frac{2((x-x_0)\sin\phi-(y-y_0)\cos\phi)\cos\phi}{b^2} \\ \frac{dP_i}{d\phi} \end{bmatrix} \quad (137)$$

$$\begin{aligned} \frac{dP_i}{d\phi} = & \frac{2((x-x_0)\cos\phi+(y-y_0)\sin\phi)(-(x-x_0)\sin\phi-(y-y_0)\cos\phi)}{a'^2} + \dots \\ & + \frac{2((x-x_0)\cos\phi+(y-y_0)\sin\phi)((x-x_0)\sin\phi+(y-y_0)\cos\phi)}{b^2} \end{aligned} \quad (138)$$

The gradient vector of the ellipse function with respect to the measurements is given by Equation 139.

$$\nabla_{\mathbf{x}} P_i = \begin{bmatrix} \frac{dP_i}{dx} \\ \frac{dP_i}{dy} \end{bmatrix} = \begin{bmatrix} \frac{2((x-x_0)\cos\phi+(y-y_0)\sin\phi)\cos\phi}{a'^2} + \frac{2((x-x_0)\sin\phi-(y-y_0)\cos\phi)\sin\phi}{b^2} \\ \frac{2((x-x_0)\cos\phi+(y-y_0)\sin\phi)\sin\phi}{a'^2} - \frac{2((x-x_0)\sin\phi-(y-y_0)\cos\phi)\cos\phi}{b^2} \end{bmatrix} \quad (139)$$

We now want to perform a simulation to show the results of the derivations for the ellipse. The following parameters are defined as the true simulation parameters.

$$a' = 5 \quad b' = 2 \quad x_0 = 3 \quad y_0 = 2 \quad \phi = \frac{\pi}{4} \quad (140)$$

The variance in  $x$  and  $y$  is the same as in the 3rd order polynomial simulation.

$$\sigma_x^2 = 0.5 \quad \sigma_y^2 = 0.75 \quad (141)$$

The generated measurements along with the true and estimated fits are shown in Figure 35. We note here that the algorithm outlined in Section 4.4.2, derived by Halir and Flusser [18] is implemented in order to determine the estimated parameters.

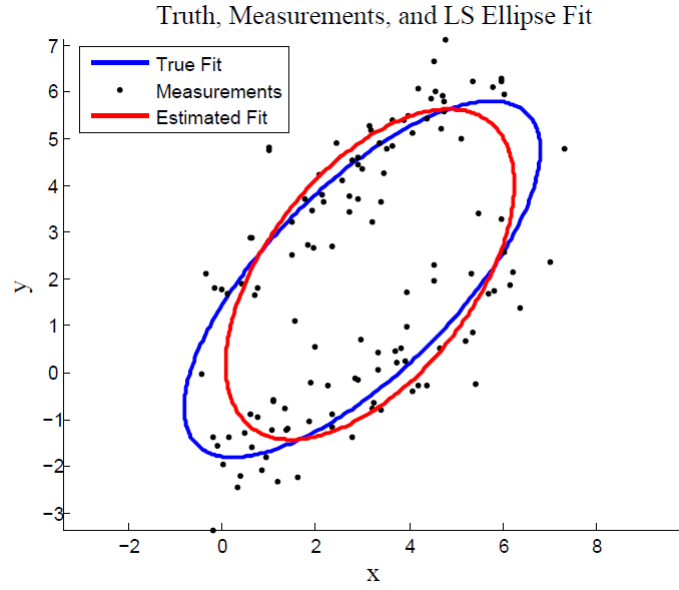


Fig. 35: Ellipse Simulation

With the generated measurements and the gradient vectors, Equation 125 is then utilized in order to determine the uncertainty in the parameter vector,  $\Theta$ . An example of the covariance matrix is presented in Equation 142.

$$\begin{aligned}
 C_{\Theta} &= \begin{bmatrix} \sigma_{a'a'}^2 & \sigma_{a'b'}^2 & \sigma_{a'x_0}^2 & \sigma_{a'y_0}^2 & \sigma_{a'\phi}^2 \\ \sigma_{b'a'}^2 & \sigma_{b'b'}^2 & \sigma_{b'x_0}^2 & \sigma_{b'y_0}^2 & \sigma_{b'\phi}^2 \\ \sigma_{x_0a'}^2 & \sigma_{x_0b'}^2 & \sigma_{x_0x_0}^2 & \sigma_{x_0y_0}^2 & \sigma_{x_0\phi}^2 \\ \sigma_{y_0a'}^2 & \sigma_{y_0b'}^2 & \sigma_{y_0x_0}^2 & \sigma_{y_0y_0}^2 & \sigma_{y_0\phi}^2 \\ \sigma_{\phi a'}^2 & \sigma_{\phi b'}^2 & \sigma_{\phi x_0}^2 & \sigma_{\phi y_0}^2 & \sigma_{\phi\phi}^2 \end{bmatrix} \\
 &= 1.0e - 003 \begin{bmatrix} 0.0042 & -0.0510 & -0.0090 & -0.0399 & -0.0201 \\ -0.0510 & 0.6239 & 0.1097 & 0.4879 & 0.2461 \\ -0.0090 & 0.1097 & 0.0193 & 0.0858 & 0.0433 \\ -0.0399 & 0.4879 & 0.0858 & 0.3815 & 0.1924 \\ -0.0201 & 0.2461 & 0.0433 & 0.1924 & 0.0970 \end{bmatrix}
 \end{aligned} \tag{142}$$



Figure 36 shows the uncertainty in each of the parameters. We have presented each possible scenario for the bounds. For example, the upper or lower bound on a single parameter,  $x_0$  is considered, and then the filled region corresponds to the upper and lower bound on the axis parameters,  $a'$  and  $b'$ . Therefore the upper and lower bounds on the axis parameters are always taken into account along with each possible combination of the remaining parameters.

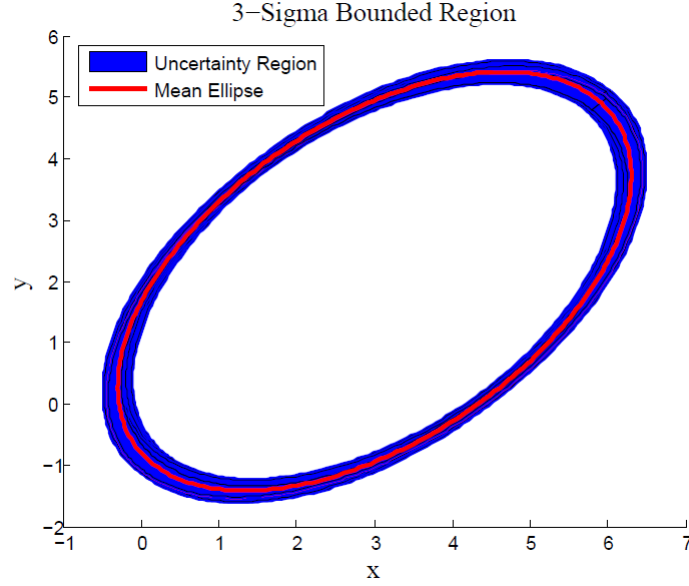


Fig. 36: Ellipse Axis Uncertainty

We again implement a Monte Carlo simulation in which the measurements are varied, the estimated ellipse fit is computed, and the covariance matrix is computed using Equation 125. The same method is used to determine the convergence characteristics of the covariance solution in which the parameters in Equation 110 are replaced with the parameters in  $\Theta$  as shown in Equation 143

$$MC_{cov} = \frac{1}{10,000} \sum_{i=1}^{10,000} \left( \begin{bmatrix} a' \\ b' \\ x_0 \\ y_0 \\ \phi \end{bmatrix} - \begin{bmatrix} \bar{a}' \\ \bar{b}' \\ \bar{x}_0 \\ \bar{y}_0 \\ \bar{\phi} \end{bmatrix} \right) \left( \begin{bmatrix} a' \\ b' \\ x_0 \\ y_0 \\ \phi \end{bmatrix} - \begin{bmatrix} \bar{a}' \\ \bar{b}' \\ \bar{x}_0 \\ \bar{y}_0 \\ \bar{\phi} \end{bmatrix} \right)^T \quad (143)$$

Where the estimated coefficients are averaged after each simulation which are denoted by,  $\bar{\hat{a}}'$ ,  $\bar{\hat{b}}'$ ,  $\bar{\hat{x}}_0$ ,  $\bar{\hat{y}}_0$ , and  $\bar{\hat{\phi}}$  respectively. Then the convergence measure is given to be:

$$convergence = |MC_{cov} - F^{-1}| \quad (144)$$

This convergence value is normalized and the result for a portion of the simulations of the is shown in Figure 37.

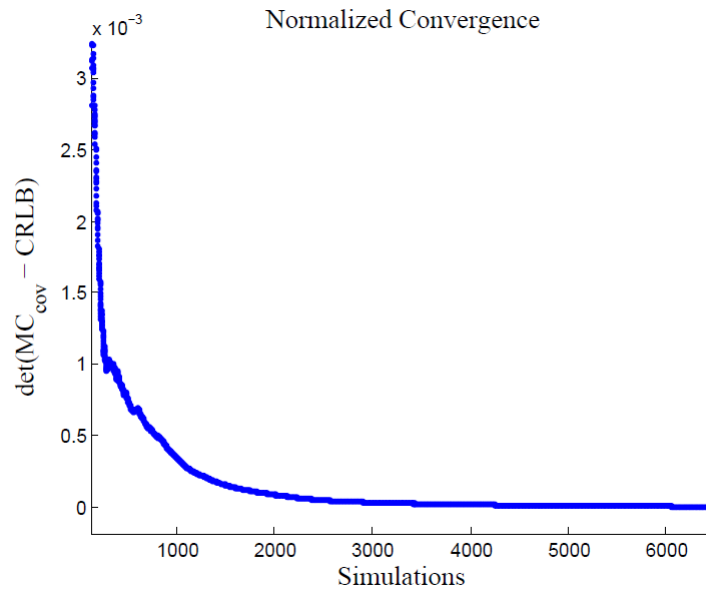


Fig. 37: Ellipse Uncertainty Convergence

As with the 3rd order polynomial fit, we also plot each of the ellipses which is defined by parameters within 3-sigma values of their respective mean's. The mean values of the parameters are:

$$\bar{a}' = 4.1178 \quad \bar{b}' = 2.3636 \quad \bar{x}_0 = 2.9998 \quad \bar{y}_0 = 1.9985 \quad \bar{\phi} = 1.0354 \quad (145)$$

Now each of the ellipses with all five of it's parameters lying within 3-sigma values of the mean is plotted in Figure 38, in blue, along with the ellipse defined by the mean values of the parameters in red.

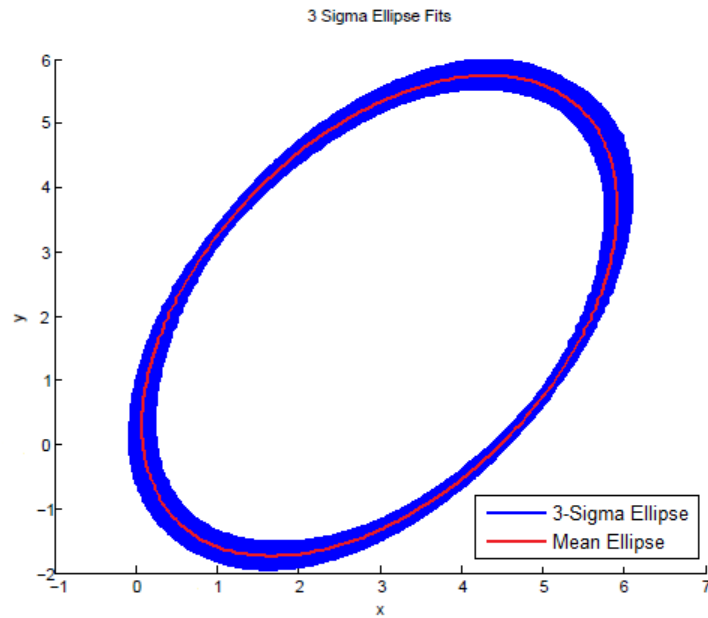


Fig. 38: 3-Sigma Ellipses

## **4.0 ASSUMPTIONS AND PROCEDURES**

Throughout the course of this section we will explain the algorithm which has been developed using the theories from the previous sections. This algorithm automatically identifies straight line segments and then through user interaction the accuracy of the finalized network can be improved. A graphical user interface (GUI) has been developed which allows the user to easily interact and manipulate the results at each step. Each of the following sections will explain the functions of all available buttons and as well as how they are implemented.

### **4.1 Overview**

The GUI consists of 13 push buttons, 7 text boxes, a single slider, a radial button box which consists of 2 buttons (i.e. only one can be active at a time) and 2 plot areas. Figure 39 depicts the layout of these features.

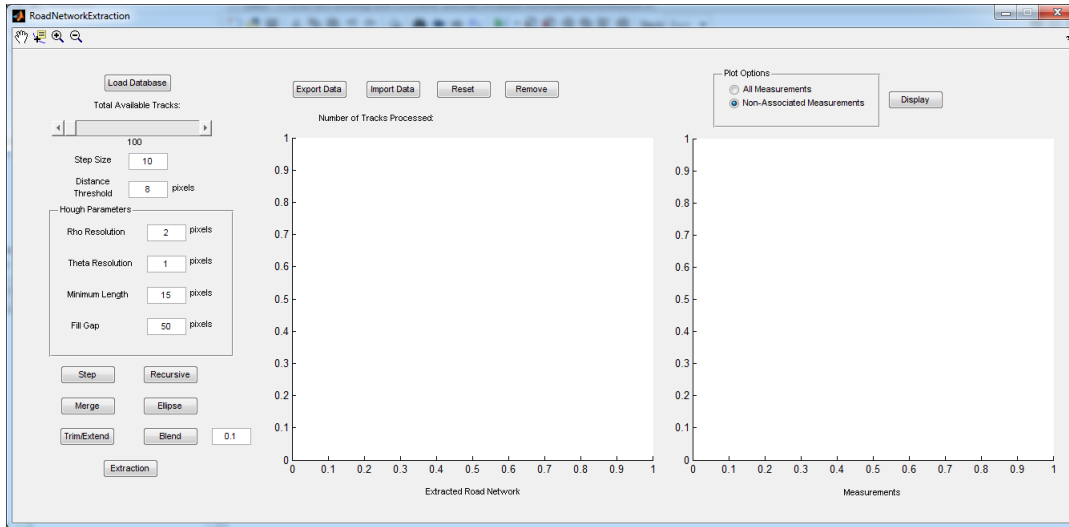


Fig. 39: Graphical User Interface

In what follows we will display fragments of the GUI which change after each operation is performed and the reader is referred to Figure 39 as a visual aide when needed. The operations will be outlined as if an actual database were being processed. For example we will start by loading the database, then extracting the lines, manipulating the network through the use of the features available within the GUI, and eventually extracting the network and storing it for later use. Figure 40 depicts a block representation of the algorithm implemented during the course of this section.

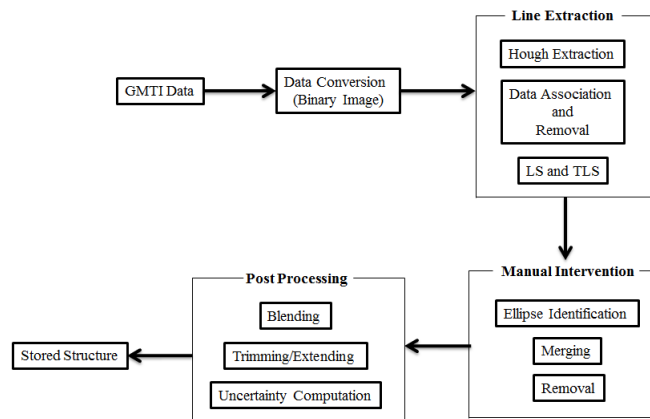


Fig. 40: Algorithm Overview

The algorithm begins by accepting a MatLab data structure which must conform to a standard set of rules.

- 1) Must be a MatLab data structure
- 2) Structure variable must begin with *trac*, neglecting case, this field should contain each of the collected tracks (i.e. *trac*  $\sim$  (1) is the first track and *trac*  $\sim$  (100) is the 100th track)
- 3) Sub-field containing the location of the collected measurements, sub-field contains *loc*, (*trac*  $\sim$  (#).*loc*  $\sim$ )
- 4) Sub-field containing the covariance matrices of each of the respective measurements, sub-field contains *cov*, (*trac*  $\sim$  .(#).*cov*  $\sim$ )

The data undergoes some preliminary manipulation which includes the conversion of the covariance matrix from meters to degrees corresponding to their respective latitudes, as well as the conversion of the measurements from latitude - longitude to pixels taking into account the size of the image (defaulted to  $1500 \times 1500$  pixels). In addition to the data conversion the data from the original structure is allocated into a second structure, *SynData*, which initializes the following sub-fields:

- *SynData.Location* - augmented matrix of all available measurements in both lat - long and pixel coordinates
- *SynData.Covariance* - augmented 3-dimensional matrix of the converted measurement covariance matrices
- *SynData.LastTrack* - the number of the last track processed
- *SynData.Xmin* - minimum of the longitude, in meters, to offset the origin in pixel coordinates
- *SynData.Ymin* - minimum of the latitude, in meters, to offset the origin in pixel coordinates
- *SynData.Origin* - the offset origin location in latitude - longitude

## 4.2 Load Database

This push button asks the user to select an appropriate database to load. The selected database must conform to several necessary rules in order to begin the processing. If either of the three required fields is missing an error box will inform the user of which field is missing. Once the data structure has been loaded there will be a slight change to the GUI. Directly under the Load Database push button, there is a string which will state how many tracks are available. In the case of the first data set we are working with, there are 4,127 tracks as shown in Figure 41.

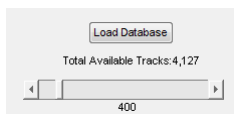


Fig. 41: Load Database Change

In addition to the noticeable change to the GUI, the slider's maximum value is updated such that it corresponds to the number of tracks available.

### 4.3 Line Extraction

This segment of the algorithm is the automated process which locates the lines in the image using the Hough Transform, determines which measurements are associated with these lines, removes the measurements and covariance matrices from their initial fields in *SynData* and stores them in the respective line estimate field. Once the association has been completed the Least Squares and Total Least Squares solutions are computed and the line's coefficient estimates are updated accordingly. Several variables for this process are obtained via the GUI. These variables include; the slider value, step size, distance threshold, and all of the parameters located within the Hough parameters box in the GUI.

#### 4.3.1 Slider

The algorithm developed is intended to be recursive such that all tracks are not readily available for the given region. However, since the test data sets provided are composed of a "complete" set of tracks the slider allows for a semi-recursive view of the data. A portion of the data can be processed by setting the slider value and then manipulated and stored. The stored data can then be loaded and the user can continue working on the data where they left off. As the slider is moved the number string below the slider changes to denote the current value.

#### 4.3.2 Step Size

This section refers to the text box to the right of the string of text, *Step Size*. The step size is a method of segmenting the data in order to process the data recursively. The default value of 10 refers to 10 tracks processed at each step. A step is performed using the push button

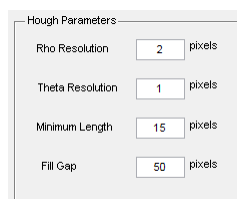
denoted by Step. This box is one of the necessary inputs prior to performing any extraction and greatly impacts the Hough Transforms extraction capabilities. Too small a number will result in too little data and lines will not be extracted as well. On the other hand, too large of a number will result in inaccurate extraction of line segments from the data (i.e. the Hough will identify undesired line segments). However, once the user has established the majority of the network and the Hough no longer identifies any segments, the user can adjust the value to a larger number in order to complete the data association more rapidly.

#### 4.3.3 Distance Threshold

This box allows the user to specify the perpendicular distance threshold used when determining the association of the data. The input should be given in a pixel measure, the default being 8 has proven to be an adequate initial value. Once the road network has been further developed the distance threshold can be increased at the user's discretion should there be data which is not associated which should be.

#### 4.3.4 Hough Parameters

In this section we will discuss the Hough Parameters box shown in Figure 42. These are the parameters which will be passed into the Hough functions in MatLab in order to extract the line segments in the image. These parameters were defined in detail in Section and will be reviewed here. Thus far the defaults for all four of the parameters have worked well and have not been altered during the course of this development. All of these parameters will refer to a pixel metric and not lat - long or meters since the Hough functions deal solely with an image.



Hough Parameters	
Rho Resolution	2 pixels
Theta Resolution	1 pixels
Minimum Length	15 pixels
Fill Gap	50 pixels

Fig. 42: Hough Parameters



1) *Rho Resolution*: The rho resolution is the quantization of the rho vector, the distance from the origin to the point. Careful consideration must be applied here as this value along with the theta resolution will determine the ability of the Hough to extract line segments appropriately.

2) *Theta Resolution*: The theta resolution is the quantization of the theta vector, the angle of the line created by the point and the x-axis. The smaller the number the larger the vector. In later versions of MatLab the Hough function has been changed. The alteration which has been implemented deals with how the theta option is handled. Rather than accepting a single number as the input and having the Hough function deal with the creation of the vector, the user is required to input an actual vector of values for theta. Therefore a version check has been incorporated in which the theta parameter issue is resolved. The theta vector remains between -90 and 90 with a quantization given by the user's input.

3) *Minimum Length*: Recall that this is the minimum allowable length of a line segment to be considered a line. If the length, given by the Euclidean distance, is less than the value defined here then the line is not extracted.

4) *Fill Gap*: The fill gap parameter is the distance between two line segments with the same rho and theta value which will be bridged if the Euclidean distance between the two nearest endpoints falls below this threshold.

#### 4.3.5 Step

The implementation of this button contains the extraction step. Prior to implementing this function the user must have changed each of the previously mentioned variables dealing with the step size, distance threshold and Hough parameters. In addition to the variable changes if desired, the user must have loaded a database to work with which conforms to the criteria presented in Section 5.1. The first set of tracks defined by the *Step Size* are passed into the algorithm and all of the measurements within these tracks are manipulated into the new *SynData* structure and converted as necessary. The conversions which occur include; converting original measurements from latitude - longitude to meters and finally to pixels and also the covariance matrix is converted from the original meters to degrees corresponding to the respective latitude position. Equations 146 to 152 depict the necessary conversion equations. The equations respectively represent, the measurement values in meters, the measurement values in pixels, the respective covariance values

in degrees according to their respective latitude.

$$X_{data_i} = R_e (Latitude(i) - Origin(1)) \quad (146)$$

$$Y_{data_i} = R_e (Longitude(i) - Origin(2)) * \cos (Origin(1)) \quad (147)$$

$$X_{index_i} = N + 1 + \text{Round} \left( \frac{X_{data_i} - X_{min}}{X_{inc}} \right) \quad (148)$$

$$Y_{index_i} = 2N - \text{Round} \left( \frac{Y_{data_i} - Y_{min}}{Y_{inc}} \right) \quad (149)$$

$$\sigma_{xx_i} = \sigma_{xx_i}^m \frac{180^2}{(R_e \cos (Latitude_i))^2} \quad (150)$$

$$\sigma_{yy_i} = \sigma_{yy_i}^m \frac{180^2}{(R_e \sin (Latitude_i))^2} \quad (151)$$

$$\sigma_{xy_i} = \sigma_{xy_i}^m \frac{180^2}{R_e \cos (Latitude_i) \sin (Latitude_i)} \quad (152)$$

In the previous equations the following variables are defined:

- $R_e$  - radius of the Earth at the equator in meters, 6,371,000 meters
- $Latitude_i$  - latitude of the  $i$ th measurement in radians
- $Longitude_i$  - longitude of the  $i$ th measurement in radians
- $Origin$  - the latitude and longitude, in radians, of the first measurement used as to shift the data
- $X_{min}$  - the minimum value of the  $X_{data}$  vector
- $Y_{min}$  - the minimum value of the  $Y_{data}$  vector
- $N$  - size of one of the image cells, default value 500
- $X_{inc}$  - the value of a single pixel in the x-direction, one pixel corresponds to 20 meters
- $Y_{inc}$  - the value of a single pixel in the y-direction, one pixel corresponds to 20 meters
- $\sigma_{xx_i}^m, \sigma_{yy_i}^m, \sigma_{xy_i}^m$  - respective covariance value in meters

In order to allow the growth of the image we pack the initial image with zeros. For the first iteration we have a tiled image of the following composition. A tiled image is more desirable as it significantly reduces the computational expense of the algorithm.

$$\begin{bmatrix} \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{Data} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} \end{bmatrix} \quad (153)$$

Where we have denoted the first set of data from the 10 tracks by **Data**. The complete image size is  $1500 \times 1500$ . Each of the nine tiles is checked for data and should there be none, the tile is ignored and the algorithm continues to the next. When data has been found in one of the tiles the Hough Transform is then implemented and extracts up to 40 lines. This number is chosen to be relatively high since the process will exit when no line segment has been found. The Hough line extraction is recursive in that a single line is identified, the measurements associated with this line segment are removed from the image and then the process repeats until no line segment is identified by the Hough. The data association requires two thresholds, one for the distance from a point to the line and the second for the distance from the endpoints. The first threshold is obtained via the distance threshold variable defined by the user, while the second threshold is set at 5. This means that the perpendicular distance from the line to a point must be less than 8 in order to be considered. If the first test is passed then the point is checked against the endpoints of the line. Should the point lie within the endpoints with the allowable tolerance of 5, the point will be associated with the line segment and removed from the image. The endpoint tolerance allows the line to grow in either direction and therefore across the cells in the image. Once the Hough has finished extracting the line segments and the image has been reduced to non-associated data the Total Least Squares algorithm outlined in Section is implemented. The algorithm first checks whether the line is oriented in a vertical or horizontal direction as this impacts the results of the Total Least Squares solution. A vertical line is identified by the Hough coefficient of the slope (the rho and theta can be transformed to slope and intercept). Should the absolute Hough slope coefficient be greater than two the algorithm is reversed and solved as a function of y for the vertically oriented line. The Least Squares solution is passed into the Total Least Squares as the initial estimate. Once the Total Least Squares algorithm has finished, the line segments as well as the remaining non-associated data are plotted in the original latitude - longitude coordinate system. This completes the processing of the 10 tracks and at this time the user can intervene and perform additional measures to modify the network and increase its accuracy. These additional features will be explained in the later sections. The results of the algorithm are shown in Figure 43 in the first plot window on the left, while the second plot area on the right can either display all of the measurements currently available or the remaining measurements which were not associated with the lines. In addition to the obvious changes in the plot areas, the text string above the left plot has been modified in order to display the amount

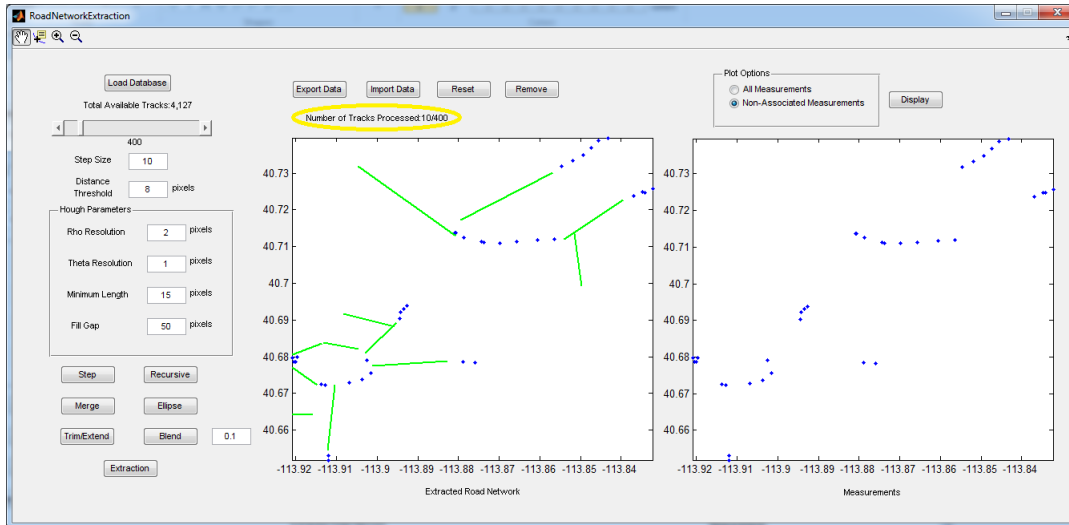


Fig. 43: 10 Tracks Processed

of processed tracks out of the total defined by the slider.

#### 4.3.6 Recursive

This button performs the exact same action as the Step button, however it will process all tracks up to the slider's current value without interruption. It will step through the data as well, for example if there are 50 tracks to be processed with a step of 10, the recursive button will process 10 tracks and show the result of the 10 tracks and then immediately continue onto the next 10 and so on until all 50 are done processing. After the 50 tracks have been processed the user can then manipulate the data as they see fit. The reason for two separate options is that with the Step button, the user has more control over the extraction process and can intervene in the processing of the data to perform additional feature extractions. The recursive function will not allow any intervention during the extraction process and the user must wait until all tracks have been processed.

## 4.4 Manual Intervention

In this section the features which pertain to the manual intervention are detailed and examples of each are provided. The three possible interventions which can be taken include merging of two similar lines based on the user's discretion, the identification of an entire ellipse, or the

removal of a previously defined feature which is undesired or not to the user's liking (i.e. a line identified by the Hough with limited data may obscure the accuracy of the overall network and can be removed to wait for additional data or an adjustment to the Hough parameters can be made).

#### 4.4.1 Merge

The first of the manual features allows the user to select two similar lines which they believe should be combined to form a single line. When the button is pressed the user will be required to select two lines by clicking near them in the left hand plot window. Each time the user selects a line, the nearest line identified will turn from the initial green color to red and the user will be prompted with a confirmation window. The confirmation window contains three options, 'Yes', 'No', and 'Cancel'. The 'Yes' option will continue the process and the user will be required to select the second line and confirm the selection again. The response 'No' will revert the selected line back to green and allow the user to re-select until they verify the validity of the choice. The third and final option, 'Cancel', will terminate the merge procedure with no changes made to the data structure. The selection of two lines to be merged is shown in Figure 44, where the lines are identified in red and the confirmation box is also shown.

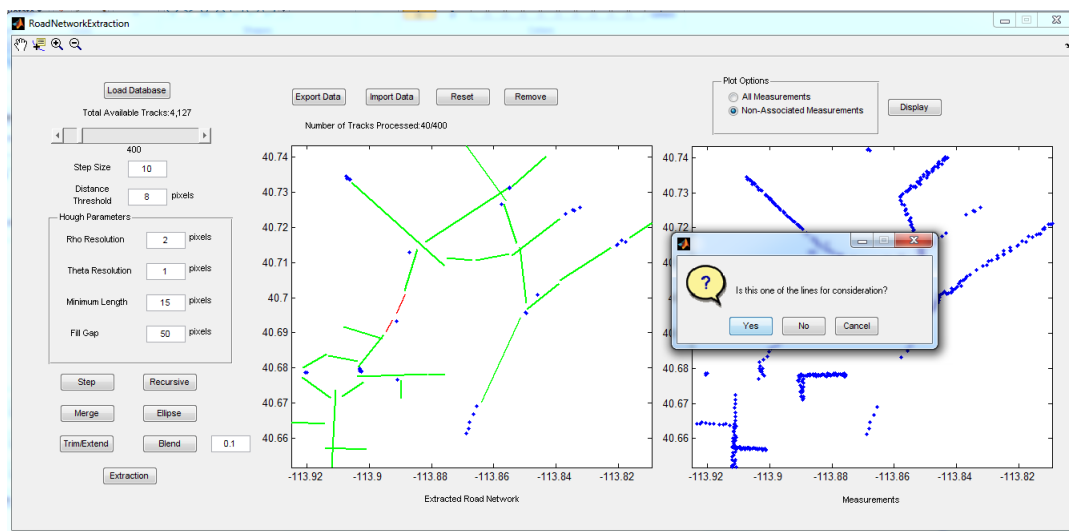


Fig. 44: Merge Selection

The algorithm will then merge the lines by combining three sub-fields corresponding to the

two line segments selected. These necessary sub-fields include; *Coordinates* (the measurements in pixel units), *LatLong* (measurements in original latitude - longitude), and *Covariance* (each of the respective covariance matrices of the measurements in degrees). Once these have been combined the Total Least Squares algorithm is used to determine the line fit of the combined measurements. The resulting merged line of the selections in Figure 44 is highlighted in Figure 45. The two lines which were initially identified are now removed from the data structure and replaced with a single field corresponding to the merged line. s

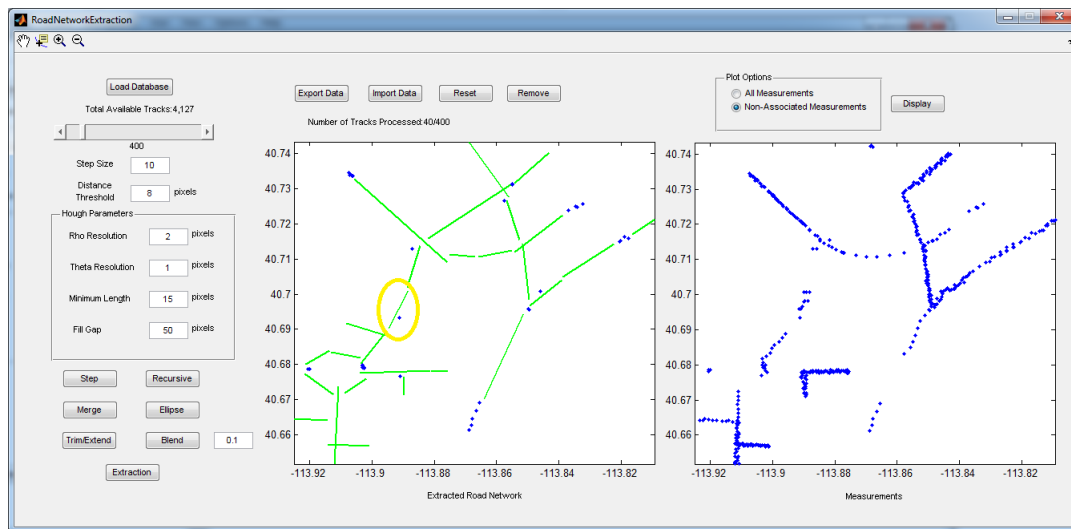


Fig. 45: Merge Results

#### 4.4.2 Ellipse

The ellipse button implements the algorithm defined by Halir and Flusser [18]. This is similar to the merge button in that the user is required to define the line segments which compose the ellipse. We note here that only a full ellipse is possible through the use of the algorithm in Section . The user is required to input the number of lines that compose the ellipse and then identify each of these lines in the same manner as with the merge algorithm. After each line is identified, the confirmation window appears, and the user makes the appropriate choice. Each of the lines is again identified by changing the color to red and once all of the lines have been identified, the ellipse algorithm takes all measurements in the *LatLong* sub-field and computes the necessary ellipse parameters. In order to compute the ellipse parameters using the measurements

in lat - long coordinates, it is necessary to normalize the measurements due to the numerical accuracy of the Least Squares ellipse algorithm. First the measurements are shifted such that they have a mean of 0 in both  $x$  and  $y$ , then the shifted values are normalized by the range of their respective axis. These shifted normalized measurements are then passed into the Least Squares ellipse algorithm. Figure 46 shows the estimated ellipse. The original lines will remain in the data structure however they will be ignored for all future operations. A sub-field in the line structure labeled, *InEllipse*, will change value from 0 to 1. This will inform the algorithm to ignore any operations pertinent to the lines (i.e. plotting, locating, and data association).

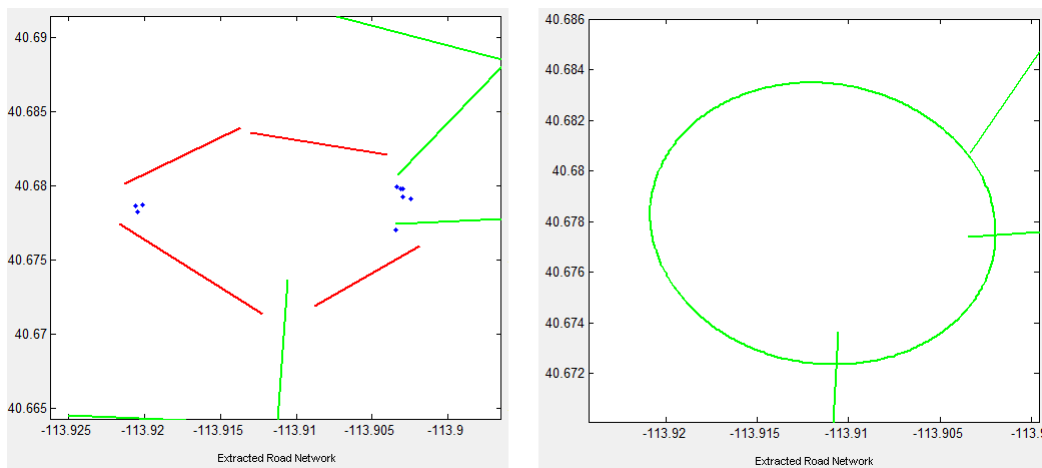


Fig. 46: Ellipse Results

The left hand plot in Figure 46 shows the four lines which were considered for the ellipse algorithm. Once the ellipse has been established, the measurements which were not associated with any line segments are re-checked to determine if they should be associated with the ellipse. This is done by first transforming the ellipse to be centered at the origin and have a rotation angle of zero. Then the remaining measurements undergo the same transformation, in their respective pixel units (easier to associate and the lines used the distance threshold of 8, ergo for consistency the same convention is used). This is done in order to simplify the optimization problem. In order to transform the coordinates we refer to Equations 154 and 155 from [24].

$$X_n = (x - x_0) \cos \phi - (y - y_0) \sin \phi \quad (154)$$

$$Y_n = -(x - x_0) \sin \phi + (y - y_0) \cos \phi \quad (155)$$

We denote the transformed points by  $(X_n, Y_n)$  and the original measurements by  $(x, y)$ . The center of the ellipse prior to transformation is  $(x_0, y_0)$  and the rotation angle of the ellipse is given by  $\phi$ . With the measurements and ellipse transformed to the new coordinate system we can more easily compute the nearest point on the ellipse which minimizes the distance to a given measurement. We will be minimizing the Euclidean distance between some arbitrary point which lies on the ellipse,  $(p, q)$  and the measurement  $(x, y)$ . The minimization of the Euclidean distance will yield the same solution as the minimization of the square of the Euclidean distance. Equation 156 represents the optimization problem to be solved, where the solution is the point which lies on the ellipse.

$$\arg \min J = (p - x)^2 + (q - y)^2 \quad (156)$$

$$\text{Subject to } \frac{p^2}{a'^2} + \frac{q^2}{b'^2} = 1 \quad (157)$$

In Equation 156 we denote the semi-axis values by  $a'$  and  $b'$  respectively. This optimization problem can be solved by the use of the Lagrange multiplier method. In this method we append the Lagrange multiplied constraint to the cost function. The Lagrange multiplier is given by,  $\lambda$ . The unconstrained objective function is then given by Equation 158.

$$\arg \min J = (p - x)^2 + (q - y)^2 + \lambda \left( \frac{p^2}{a'^2} + \frac{q^2}{b'^2} - 1 \right) \quad (158)$$

In order to solve Equation 158 we must differentiate with respect to  $p$  and  $q$ , in addition to considering the constraint. Performing the necessary calculations and simplifying the equations



by solving for  $p$  and  $q$  we obtain the following system of equations.

$$\frac{dJ}{dp} : p = \frac{x^2}{a'^2 + \lambda} \quad (159)$$

$$\frac{dJ}{dq} : q = \frac{yb'^2}{b'^2 + \lambda} \quad (160)$$

$$a'^2b'^2 - p^2b'^2 - q^2a'^2 = 0 \quad (161)$$

Substituting Equations 159 and 160 into Equation 161 results in Equation 162. In addition to the substitutions we remove the fractions from the equations.

$$(a'^2 + \lambda)^2 (b'^2 + \lambda)^2 \left( a'^2b'^2 - \frac{x^2a'^4b'^2}{(a'^2 + \lambda)^2} + \frac{y^2a'^2b'^4}{(b'^2 + \lambda)^2} = 0 \right) \quad (162)$$

Multiplying and reducing Equation 162 results in a fourth order polynomial in  $\lambda$ . The coefficients of this polynomial are as follows.

$$\lambda^4 : 1 \quad (163)$$

$$\lambda^3 : 2a'^2 + 2b'^2 \quad (164)$$

$$\lambda^2 : b'^4 + 4a'b'^2 + a'^4 - a'^2x^2 - b'^2y^2 \quad (165)$$

$$\lambda^1 : 2a'^2b'^4 + 2a'^4b'^2 - 2a'^2b'^2x^2 - 2a'^2b'^2y^2 \quad (166)$$

$$\lambda^0 : a'^4b'^4 - a'^2b'^4x^2 - a'^4b'^2y^2 \quad (167)$$

This solution will result in four possible values of  $\lambda$ , two to four of the solutions will be imaginary since a line can only intersect an ellipse at two points (a single point if it is tangent). Therefore, substituting the real solutions back into Equations 159 and 160 will yield the potential points on the ellipse. Then we simply check the distances to these points and the minimum distance yields the nearest point.

#### 4.4.3. Remove

This button simply removes the selected feature and restores the measurements which were associated with the feature. Figure 47 shows the removal of the ellipse which was generated and shown in Figure 46. The removal of the ellipse requires multiple steps. This is due to the original ellipse being composed of line segments. When the user selects the ellipse, the original line segments will be restored. Then should the user wish to remove these segments they may do so by again selecting the remove button and individually selecting each line, as was done in Figure 47.

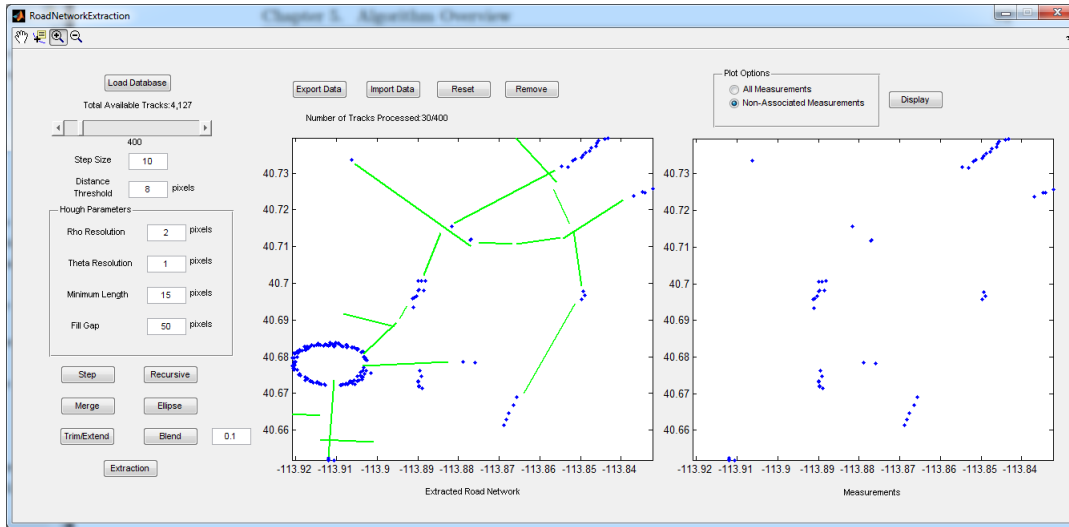


Fig. 47: Removal of Ellipse

## 4.5 Post Processing

The post processing box in 40 refers to the functions which should be implemented once all of the available data has been received and processed. These include; the use of the blending function which allows the user to select two lines and create a third order polynomial between the two lines, the trim and extend button which allows the user to modify the limits of a feature with respect to a second feature, the computation of the Cramer Rao lower bounds estimate for each of the existing features (lines, ellipses, and third order polynomials), and finally the extract feature which allows the association and extraction of additional lines using the remaining un-associated measurements. Each of these features is explored in the next sections.

### 4.5.1 Blend

This button enables the user to two lines, in the same manner as with the merge option, and create a 3rd order polynomial, between the selected lines, in  $x$  as defined by Equation 168.

$$y = Ax^3 + Bx^2 + Cx + D \quad (168)$$

The box next to the blend button which is defaulted at 0.1 denotes the percentage, 10%, of the lines to cut and consider in the polynomial estimation. Once the user has selected and confirmed the two line segments the algorithm will then determine the two nearest endpoints and trim the

specified percentage on each line. The new endpoints will then also correspond to the endpoints of polynomial. Figure 48 shows the polynomial blending function applied to two lines identified in red. We note here that the polynomial is not updated when new data becomes available.

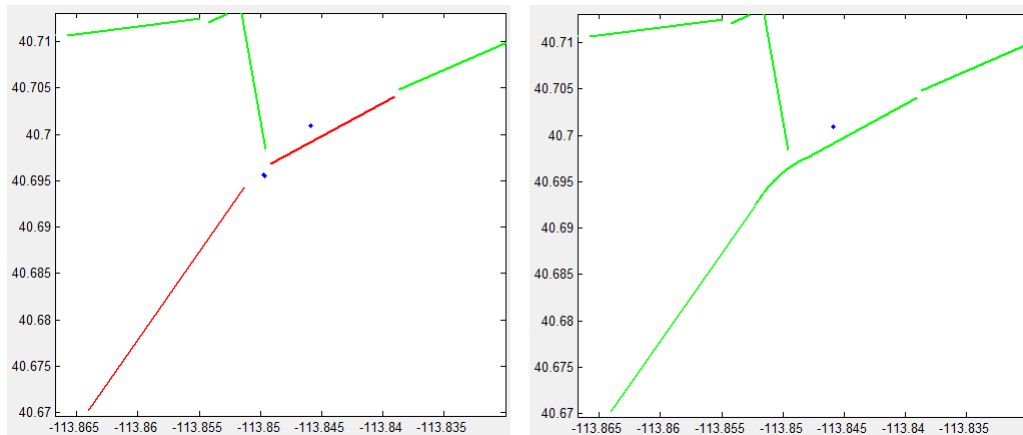


Fig. 48: Polynomial Blend

Therefore it is ideal to utilize the blending after all processing has been completed otherwise the polynomials will not connect with the line estimates since the line estimates are continuously updated.

#### 4.5.2 Trim/Extend

The Trim/Extend button allows for a complete and smooth road network. This function works in a particular order and only certain cases have been considered. First of all the user must select two geometric features, which must be confirmed as in the previous algorithms. The first feature selected will be trimmed/extended with respect to the second feature. The following cases have been considered:

- Line to Line - algorithm computes the intersection of the two lines and then determines the minimum distance of the endpoints of the first line to the intersection point. The point which is nearer is then replaced with the intersection point.
- Line to Curve - the algorithm computes the possible intersection points of the line with the selected 3rd order polynomial. The distance between both endpoints and each of the potential intersection points is computed and again the nearest intersection/endpoint combination is identified and replaced.
- Line to Ellipse - here the algorithm computes the nearest intersection point between the line and the selected ellipse and trims/extends the line to the intersection. If the line already intersects the ellipse then the endpoint which has passed through the ellipse is removed and replaced with the intersection.
- Curve to Line - the same as Line to Curve except in this case the Curve will be altered
- Curve to Curve - the intersection of two 3rd order polynomials is determined and trimmed accordingly

Each of the cases explained are presented in Figure 49.

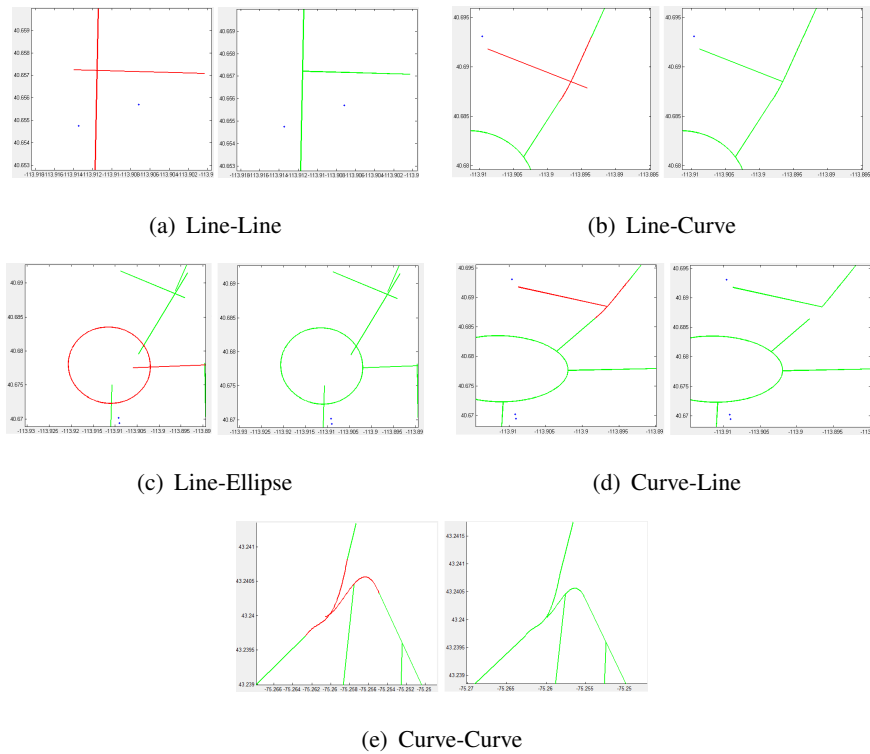


Fig. 49: Trimmed/Extended Features

#### 4.5.3 Uncertainty Computation

There is no separate button which allows one to compute the Cramer Rao lower bounds estimates of the extracted features. Instead, when the created data structure is saved using the *Export* button, the user is asked whether or not they wish to compute the estimate for the Cramer Rao bounds. Due to the number of measurements, displaying the bounds for the ellipse in a graphical manner is not applicable since as the number of measurements grows the uncertainty in the estimate tends to zero, this can be seen in the summation in the Fisher Information Matrix. The polynomial blends which were created have a similar problem, where there are either a large amount of measurements or very few measurements due to the range of the polynomial. The line uncertainty was computed in a series of steps which results in the final calculation of the uncertainty in the  $x$  and  $y$  space.

- 1) The data was shifted to obtain a mean of zero in both the latitude and longitude.
- 2) Initial guess for the TLS algorithm using the previous slope and zero for the intercept.

- 3) Calculation of the CRLB using Equations 101 through 106.
- 4) Jacobian transformation to obtain the error in the estimates of  $x$  and  $y$  given by Equations 116 through 119.
- 5) Addition of the measurement's covariance matrix.

#### 4.5.4 Extract

This feature allows the user to only extract and associate data. The importance of this feature is such that if the user has already begun working on blending, trimming and extending line segments, using the Step button will alter the lines back to the original endpoints and the user will lose the work done. The extract button allows only new lines to be extracted, in the case the user has removed existing features and wishes to modify the Hough Parameters in order to detect smaller line segments or with different orientation. In addition to extracting new segments this can also associate data, if the distance threshold is altered. Measurements can be associated with a polynomial blending function based on the nearest point on the curve which is obtained by solving the minimization of Equation 169, where the measurement is given to be  $(x, y)$  and the point on the polynomial can be defined as  $(x_p, F(x_p, A, B, C, D))$ .

$$Dist = \sqrt{(x - x_p)^2 + (y - F(x_p, A, B, C, D))^2} \quad (169)$$

Differentiation of Equation 169 yields:

$$\frac{dDist}{dx_p} = \frac{-2(x - x_p) - (y - (Ax_p^3 + Bx_p^2 + Cx_p + D))(3Ax_p^2 + 2Bx_p + C)}{2\sqrt{(x - x_p)^2 + (y - (Ax_p^3 + Bx_p^2 + Cx_p + D))^2}} \quad (170)$$

Expanding Equation 170 and organizing, results in the following equation:

$$3A^2x_p^5 + 5ABx_p^4 + (4AC + 2B^2)x_p^3 + (3BC + 3AD - 3Ay)x_p^2 + \dots \quad (171)$$

$$(C^2 + 2CD - 2By + 1)x_p + (CD - Cy - x) = 0 \quad (172)$$

Using the roots command in MatLab gives the point which lies on the polynomial and is closest to the measurement in question. We then simply check the distance and should it lie within the specified range, it is associated with the polynomial and removed from the image.

## 4.6 Stored Structure

The final box in Figure 40 is the storing of the processed data. Throughout the algorithm a MatLab data structure has been created and continuously altered in order to accommodate the necessary features and data associated with such features. The saving of this structure is done using the *Export* button near the top of the GUI.

### 4.6.1 Export Data

This button will performs two tasks. The first function herein computes the Cramer Rao lower bounds estimates for each of the extracted features. The second function is to allow the user to store all the work done during the extraction process in a MatLab data structure. The structure is outlined as follows with the field names and sub-fields labelled accordingly.

- *SynData* - this is the title of the data structure.
  - *SynData.Location* - the first field which stores the measurements in a  $N \times 4$  matrix. The first two columns correspond to the original latitude and longitude coordinates and the last two columns correspond to the indexed  $y$  and  $x$  values respectively. Only the non-associated measurements remain in this matrix.
  - *SynData.LastTracks* - the number of the last track processed.
  - *SynData.Covariance* -  $2 \times 2 \times N$  matrix of covariance matrices corresponding to each of the non-associated measurements.
  - *SynData.Ellipse* - field of all of the identified ellipses, contains the following sub-fields:
    - \* *SynData.Ellipse(#).Coordinates* - contains the associated pixel measurements.
    - \* *SynData.Ellipse(#).LatLong* - latitude and longitude measurements.
    - \* *SynData.Ellipse(#).Covariance* - measurement covariances.
    - \* *SynData.Ellipse(#).EllipseCoeffs* - ellipse coefficients in terms of parameters,  $x_0$ ,  $y_0$ ,  $a'$  and  $b'$ .
    - \* *SynData.Ellipse(#).Phi* - rotation angle of the ellipse.
    - \* *SynData.Ellipse(#).Xtran* - transformed  $x$  values to be oriented with zero rotation angle and centered at the origin in terms of the pixel coordinate system.
    - \* *SynData.Ellipse(#).Ytran* - transformed  $y$  values to be oriented with zero rotation angle and centered at the origin in terms of the pixel coordinate system.

- \* *SynData.Ellipse(#).CRLB* - Cramer Rao lower bounds estimate matrix pertaining to each of the parameters defining the uncertainty.
- *SynData.PolyBlend* - field of the blend functions with corresponding sub-fields:
  - \* *SynData.PolyBlend(#).Coordinates* - contains the measurements associated with the polynomial.
  - \* *SynData.PolyBlend(#).LatLong* - measurements associated with the polynomial segment in terms of latitude longitude.
  - \* *SynData.PolyBlend(#).Covariance* - covariance matrices corresponding to the associated measurements.
  - \* *SynData.PolyBlend(#).Coeffs* - polynomial coefficients from the *polyfit* function.
  - \* *SynData.PolyBlend(#).Xspace* - range over which the polynomial exists.
  - \* *SynData.PolyBlend(#).Fvalues* - y values for the polynomial corresponding the xspace.
  - \* *SynData.PolyBlend(#).PixelCoeffs* - coefficients for the polynomial blend corresponding to the pixel space. Used in data association.
  - \* *SynData.PolyBlend(#).PixelEnds* - endpoints of the polynomial in the pixel space, used in data association.
  - \* *SynData.PolyBlend(#).CRLB* - the Cramer Rao lower bounds estimate corresponding to the polynomial coefficients.
  - \* *SynData.PolyBlend(#).CRLB\_xy* - the Cramer Rao lower bounds estimate converted to the x-y space using the Jacobian transformation and adding the original measurement covariance.
- *SynData.Xmin* - stores the reference origin's x value from the first iteration.
- *SynData.Ymin* - stores the reference origin's y value from the first iteration.
- *SynData.Origin* - the latitude longitude location of the origin, the first measurement received is considered the origin.
- *SynData.LineEstimate* - extracted line segments with sub-fields listed.
  - \* *SynData.LineEstimate(#).Coordinates* - measurements associated with the line segment in terms of pixels.
  - \* *SynData.LineEstimate(#).LatLong* - measurements associated with the line segment



in terms of latitude longitude.

- \* *SynData.LineEstimate(#).Covariance* - covariance matrices pertaining to the associated measurements.
- \* *SynData.LineEstimate(#).Endpoints* - the endpoints determined by the Hough transformation, in pixels.
- \* *SynData.LineEstimate(#).Rho* - Hough parameter corresponding to the line.
- \* *SynData.LineEstimate(#).Theta* - Hough parameter corresponding to the line.
- \* *SynData.LineEstimate(#).Hough Coefficients* - coefficients of the line converted from rho and theta to slope and intercept.
- \* *SynData.LineEstimate(#).InEllipse* - this tells several parts of different algorithms whether the line is found in the ellipse field.
- \* *SynData.LineEstimate(#).LineCoefficients* - the estimated slope and intercept from the TLS algorithm.
- \* *SynData.LineEstimate(#).TrueLat* - estimated values of latitude, output from the TLS algorithm.
- \* *SynData.LineEstimate(#).TrueLong* - estimated values of longitude, output from the TLS algorithm.
- \* *SynData.LineEstimate(#).EstimateEndpoints* - values output from the TLS algorithm differ from the Hough function's output.
- \* *SynData.LineEstimate(#).F* - Fisher Information Matrix.
- \* *SynData.LineEstimate(#).CRLB* - Cramer Rao lower bounds matrix estimate for the uncertainty in the line parameters.
- \* *SynData.LineEstimate(#).CRLB\_xy* - the Cramer Rao lower bounds estimate converted to the x-y space using the Jacobian transformation and adding the original measurement covariance.

## 4.7 Additional Features

There are a few additional features in the GUI which perform a few necessary tasks such as resetting the GUI to the original state, allowing the user to import a pre-existing data structure and the manipulation of the right hand plot.

#### 4.7.1 Reset

As the button suggests, this will remove all data from the GUI and allow the user to begin again from scratch. The default values of each of the text boxes is also restored.

#### 4.7.2 Import Data

This feature will allow the user to import an existing data structure which was exported by the GUI and allow the user to continue working on the extraction or introduce new data. Once the user has imported the data structure they can then load a database (via the Load Database button) and continue the extraction process. The criteria of the loaded database remains the same as previously defined in addition to the criteria that the algorithm will automatically prevent the user from using any tracks previously utilized. For example if the first database's track number ended at 500 then the second database's first track number must start at 501 or else the algorithm will not allow the previous tracks to be used.

#### 4.7.3 Image Options

The Image Options are attributed to the second plot area in the GUI. The *All Measurements* option will display all measurements from all tracks currently being processed while the *Non-Associated Measurements* will display only the measurements which have not been associated with any of the existing features. Both options are shown in Figure 50

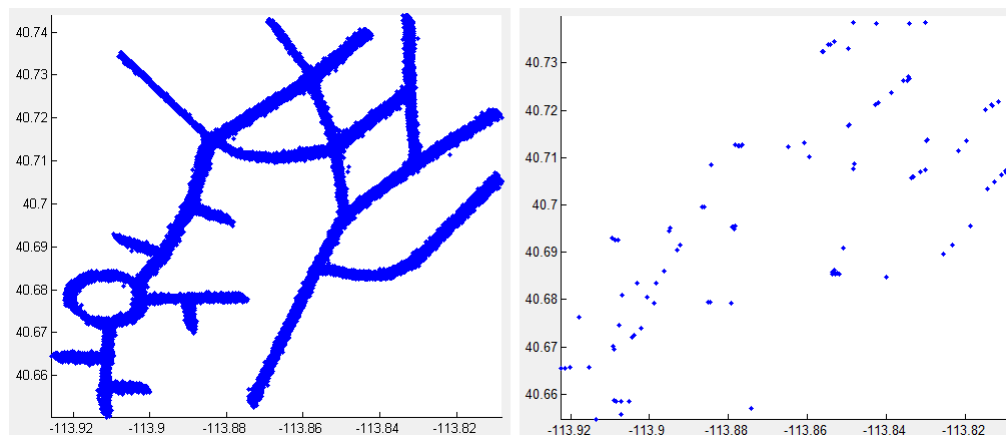


Fig. 50: Image Options

## 5.0 RESULTS

In this section we import two data structures available for the purposes of testing the capabilities of the implemented algorithm. The structure of the supplied data is examined and the measurements are plotted prior to any transformations. Each structure is then loaded into the GUI from Section 4.0 and the extraction process is implemented. The final results of the algorithm in graphical forms corresponding to the lat - long coordinate system are shown. In addition to the graphical representation of the network, examples of the numerical results pertaining to the Cramer Rao lower bounds estimate for a single case of each feature are presented.

### 5.1 Synthetic Data

The data which has been supplied consists of simulated GMTI tracks generated by the Air Force Research Labs in Rome, NY. Each track varies in the number of measurements it contains, however the structure of each track is consistent. The data structure is broken down as follows:

- *tracks* - main field of the structure.
  - *tracks(#).loc* -  $M \times 2$  matrix of latitude and longitude coordinates.
  - *tracks(#).cov* -  $4 \times 4 \times M$  covariance matrices corresponding to each measurement of the form:
 
$$\begin{bmatrix} \sigma_{xx}^2 & \sigma_{xy}^2 & \sigma_{x\dot{x}}^2 & \sigma_{x\dot{y}}^2 \\ \sigma_{yx}^2 & \sigma_{yy}^2 & \sigma_{y\dot{x}}^2 & \sigma_{y\dot{y}}^2 \\ \sigma_{\dot{x}x}^2 & \sigma_{\dot{x}y}^2 & \sigma_{\dot{x}\dot{x}}^2 & \sigma_{\dot{x}\dot{y}}^2 \\ \sigma_{\dot{y}x}^2 & \sigma_{\dot{y}y}^2 & \sigma_{\dot{y}\dot{x}}^2 & \sigma_{\dot{y}\dot{y}}^2 \end{bmatrix}$$
  - *tracks(#).vel* -  $M \times 2$  matrix of component velocities at the same time the measurement is taken.
  - *tracks(#).update* - unix time representation of measurement time (seconds since January 1, 1970).

The two supplied data structures follow the same format with one exception. The second data set's covariance array contains an extra matrix due to an error in the initialization of the array. This extra matrix corresponds always to the first matrix and is simply a matrix of zero values, which is removed during the pre-processing stage of the algorithm. The first data structure consists of 4,127 tracks and the second contains 1,675, where each track contains a varying

amount of measurements. In Figures 51 and 52 we show the plotted measurements from each set respectively.

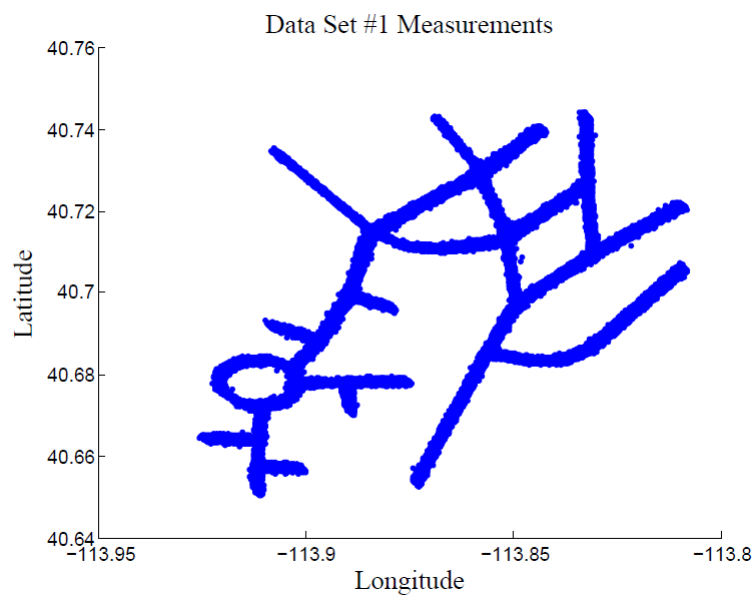


Fig. 51: Data Set #1 Measurements

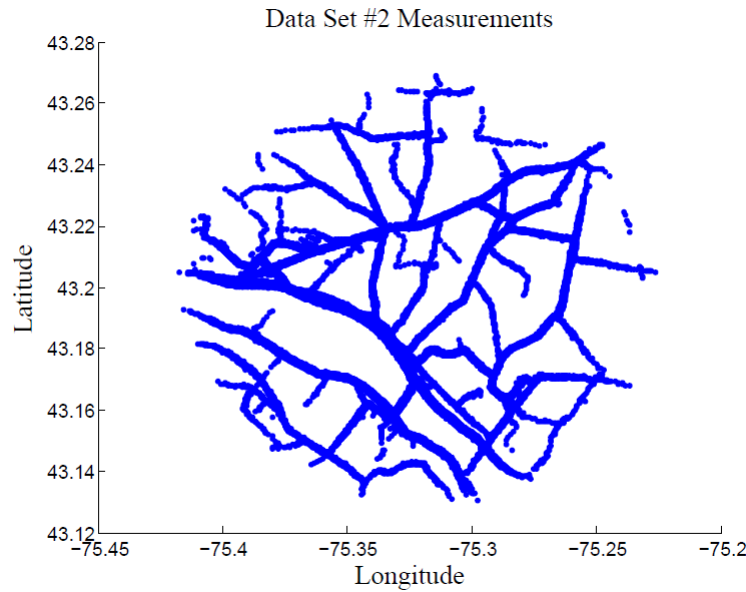


Fig. 52: Data Set #2 Measurements

## 5.2 Data Set #1 Results

In this section we process the first available data set which contains 4,127 tracks with a total of 106,936 measurements. The GUI described in Section 4.0 allows us to alter the desired variables pertaining to the line extraction if need be. With the first data set the extraction process took merely four steps:

- 1) An initial processing of approximately 100 tracks to provide a rough estimate of the overall network along with the identification of the ellipse.
- 2) Processing of remaining measurements, 4,027 additional tracks processed 10 at a time with no user intervention.
- 3) Merging, blending, trimming, extending, and removal of appropriate line segments.
- 4) Final association of remaining data by increasing the distance threshold from 8 to 12.

In the next subsection we show the graphical results in the step by step manner and then present an example of the Cramer Rao lower bounds estimate for one set of extracted features' parameters (i.e. a line segment, the ellipse, and a polynomial).

### 5.2.1 Graphical Results

The first step was to process enough tracks until a suitable number of lines were extracted and an overall estimate of the network has been obtained. This took around 100 tracks and Figure 53 shows the rough estimate of the overall network. We note that it is not required to obtain a complete representation of the network, however it can speed up the processing of the tracks.

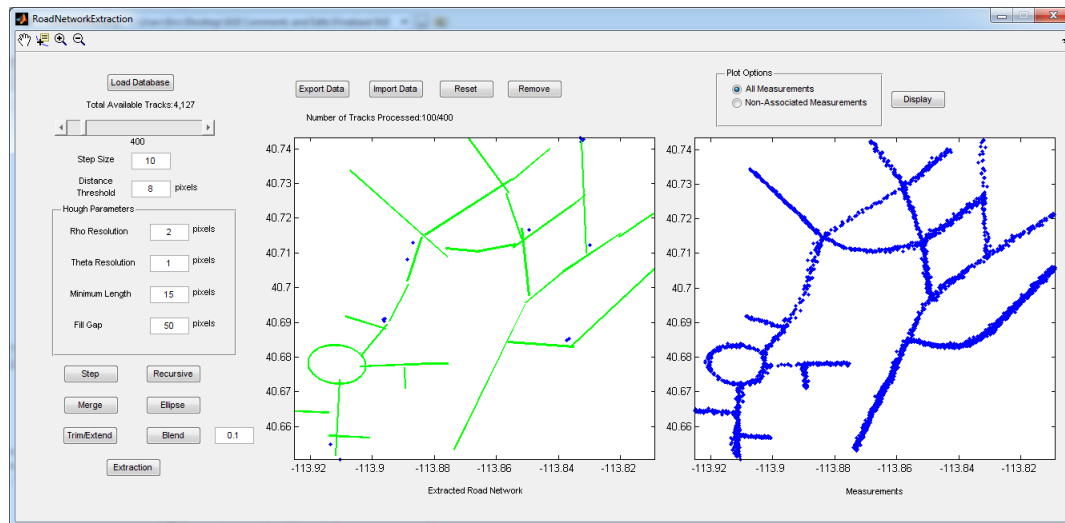


Fig. 53: Data Set #1 Phase One

This stage took only a few minutes to complete and required one ellipse identification composed of four lines and six line merging operations. Once this was complete the remaining data was processed simultaneously. The slider was forced all the way to the right to account for all 4,027 remaining tracks and the step was increased to the number of remaining tracks in order to allow the algorithm to accept all measurements in the remaining tracks. This process took approximately 30 minutes to complete using all of the default parameters set by the GUI. Figure 54

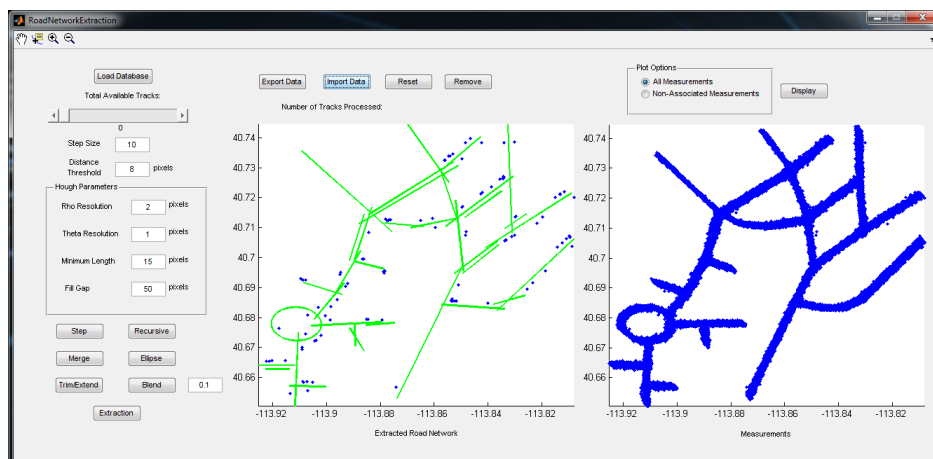


Fig. 54: Data Set #1 Phase Two

As can be seen in Figure 54, due to the distance threshold, several lines were identified which should be merged with similar nearby line segments. These lines were quickly merged to produce Figure 55.

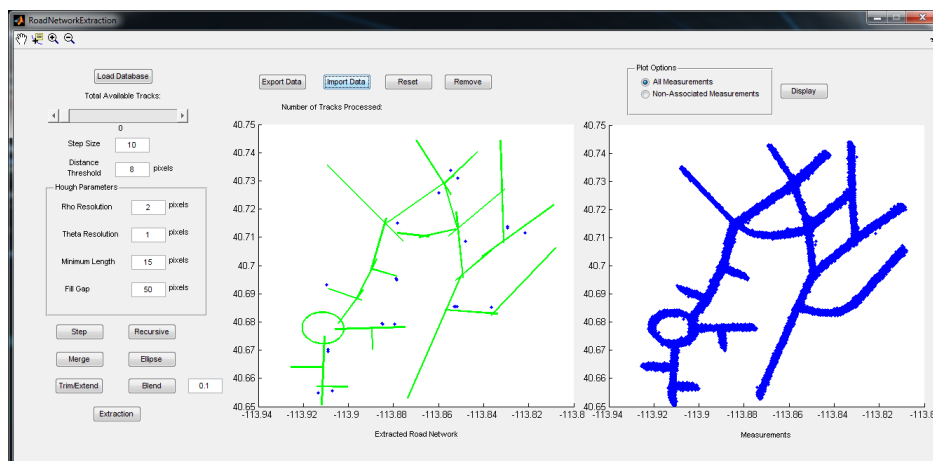


Fig. 55: Data Set #1 Phase Three

The final stage is to use the blending and trim/extend functions to produce a fluent road network which is considered to be complete based on the received measurements. The blending function's distance percentage which was trimmed from each of the two lines varied from 10% to 25% for different scenarios. Generally a higher percentage allowed for a longer less curved blend which was desirable in some cases. Figure 56 shows the final extraction with all of the appropriate manual interactions implemented.

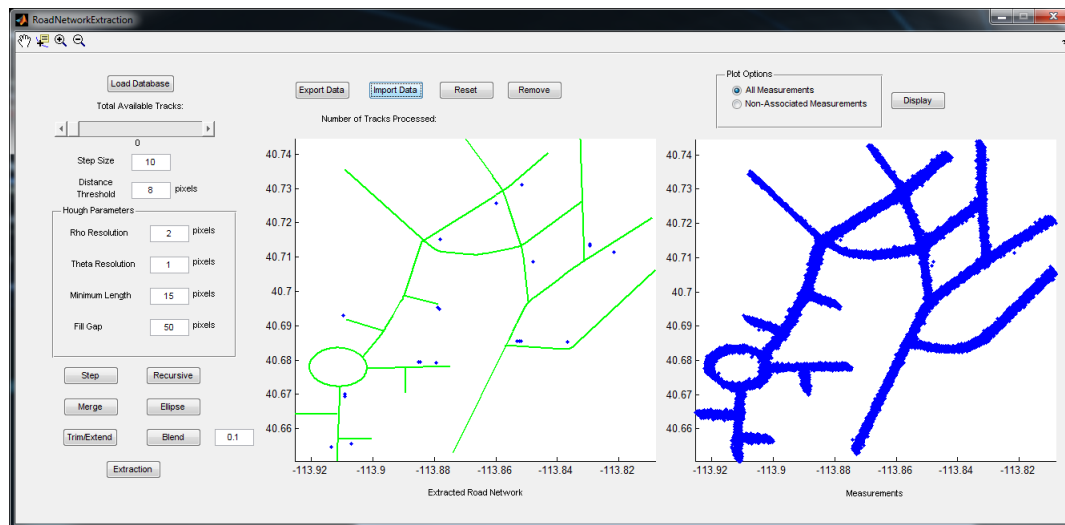


Fig. 56: Data Set #1 Final Extraction Results

The final results of the extraction is 29 line segments, a single ellipse and 12 polynomial blending functions. Once all of the tracks had been processed it took about 15 minutes to complete the manual blending, trimming, extending and merging processes on the extracted features. Therefore to process this entire data set approximately 45 minutes was required. A total of 106,915 out of the 106,936 total measurements have been associated with existing features in the road network, this corresponds to a 99.98% data association.

### 5.2.2 Uncertainty Analysis

Here we present the results for the uncertainty pertaining to each of the extracted features using the equations derived in Section 3.11. The Cramer Rao lower bounds estimates for each of the features converges to very small values due to the summation across each of the measurements associated with the feature. For example, a typical line contains anywhere from 2,000 to 4,000



measurements while the ellipse identified in this data set contains just over 6,000 measurements and the polynomials contain on average 1,500 measurements. Here we will present the results for the uncertainty in each of the estimates with numerical examples and for the linear uncertainty we display the Cramer Rao lower bounds graphically. The linear uncertainty can be transformed using the Jacobian transformation to obtain the uncertainty in terms of  $x$  and  $y$  and supplement these with their original measurement covariance matrices. Figure 57 shows the Cramer Rao lower bounds uncertainty in the estimates of the line's parameters in red, which as can be seen these bounds are very tight.

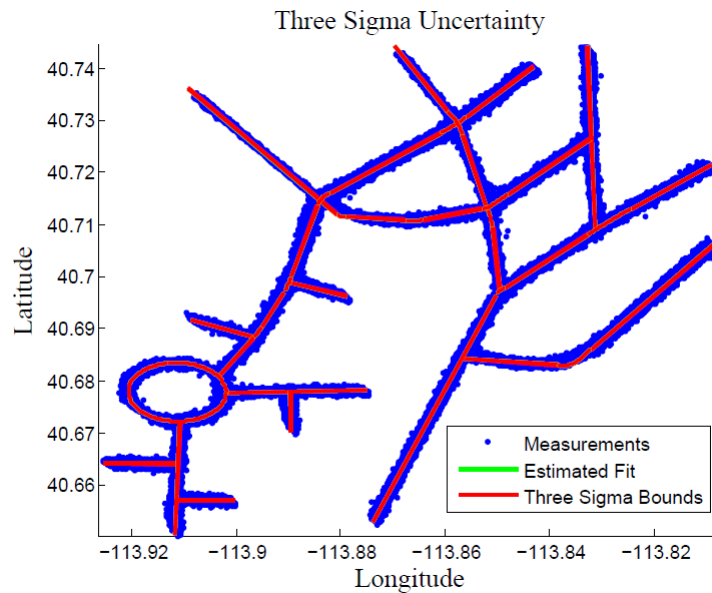


Fig. 57: Data Set #1 Line CRLB

We now supplement the figure with a numerical example of the Cramer Rao lower bounds estimate corresponding to each of the parameter estimates. This line has 4,387 measurements associated with it.

$$\begin{bmatrix} \sigma_{mm}^2 & \sigma_{mb}^2 & \sigma_{mx_t}^2 \\ \sigma_{bm}^2 & \sigma_{bb}^2 & \sigma_{bx_t}^2 \\ \sigma_{x_tm}^2 & \sigma_{x_tb}^2 & \sigma_{x_tx_t}^2 \end{bmatrix} = \begin{bmatrix} 1.7211e-06 & 3.5177e-10 & -2.5080e-10 \\ 3.5177e-10 & 1.8025e-10 & -1.2555e-10 \\ -2.5080e-10 & -1.2555e-10 & 1.3295e-10 \end{bmatrix} \quad (173)$$

Using the Jacobian transformation we can obtain the Cramer Rao lower bounds in terms of  $x$  and  $y$ , this is done using the measurements shifted such that they have a mean of zero, since

this is how the bounds were computed. In addition the Jacobian transformed CRLB is added to the original covariance matrix of the measurements. An example of a measurement covariance matrix is given by Equation 174.

$$\begin{bmatrix} \sigma_{xx}^2 & \sigma_{xy}^2 \\ \sigma_{yx}^2 & \sigma_{yy}^2 \end{bmatrix} = 1.0e - 05 \times \begin{bmatrix} 0.0793 & -0.1187 \\ -0.1187 & 0.2216 \end{bmatrix} \quad (174)$$

The Jacobian transformed CRLB is added to the measurement covariance defined in Equation 174 to produce the final covariance in the measurement including the covariance in the estimate given by Equation 175.

$$\begin{bmatrix} \sigma_{xx}^2 & \sigma_{xy}^2 \\ \sigma_{yx}^2 & \sigma_{yy}^2 \end{bmatrix} = 1.0e - 05 \times \begin{bmatrix} 0.0793 & -0.1187 \\ -0.1187 & 0.2217 \end{bmatrix} \quad (175)$$

Although there is minimal change in the covariance, the measurement covariance is on the order of 1e-5 while the transformed CRLB tends to be of the order of 1e-9, since we are dealing with latitude and longitude values a change of 1e-9 is still noticeable in the data. We now present a single line estimate from the first data set with several measurements. Figure 58 shows the line estimate in green, several measurements in blue, and each measurement's error ellipse corresponding to one sigma value.

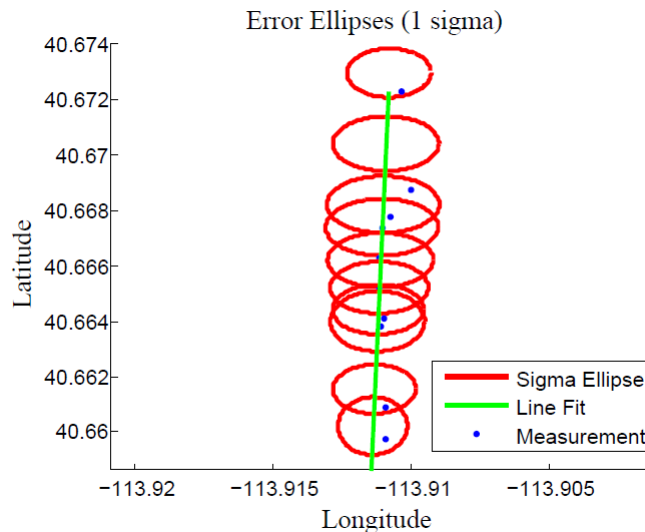


Fig. 58: Data Set #1 Error Ellipses One Sigma

Using the error ellipses we compute the mean covariance in  $x$  and  $y$ . This covariance is then

used to represent a mean error in the line using one sigma we can see from Figure 59 that this encompasses the majority of the data associated with each of the line estimates.

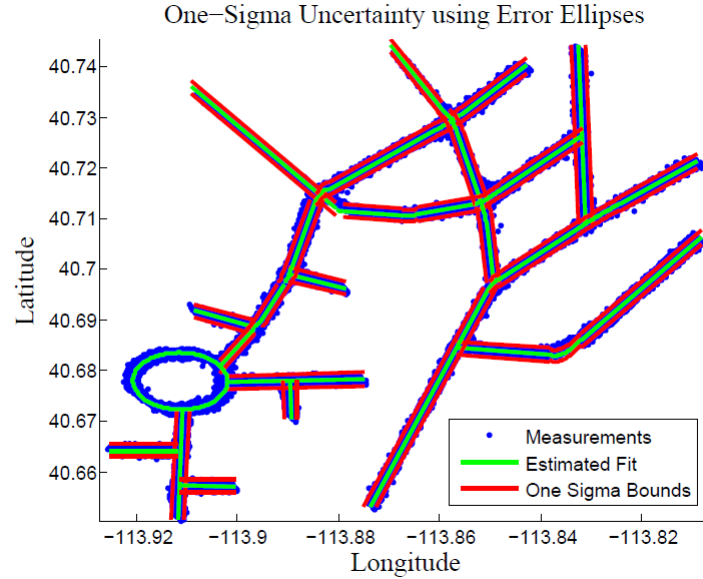


Fig. 59: Data Set #1 Mean Line Uncertainty

The ellipse estimate contains 6,816 measurements which causes the CRLB estimate to converge to a very finite value, therefore we simply present the numerical results here in Equation 176.

$$\begin{aligned}
 & \begin{bmatrix} \sigma_{a'a'}^2 & \sigma_{a'b'}^2 & \sigma_{a'x_0}^2 & \sigma_{a'y_0}^2 & \sigma_{a'\phi}^2 \\ \sigma_{b'a'}^2 & \sigma_{b'b'}^2 & \sigma_{b'x_0}^2 & \sigma_{b'y_0}^2 & \sigma_{b'\phi}^2 \\ \sigma_{x_0a'}^2 & \sigma_{x_0b'}^2 & \sigma_{x_0x_0}^2 & \sigma_{x_0y_0}^2 & \sigma_{x_0\phi}^2 \\ \sigma_{y_0a'}^2 & \sigma_{y_0b'}^2 & \sigma_{y_0x_0}^2 & \sigma_{y_0y_0}^2 & \sigma_{y_0\phi}^2 \\ \sigma_{\phi a'}^2 & \sigma_{\phi b'}^2 & \sigma_{\phi x_0}^2 & \sigma_{\phi y_0}^2 & \sigma_{\phi\phi}^2 \end{bmatrix} \\
 = & \begin{bmatrix} 1.7503e-12 & -1.5146e-13 & 9.5198e-13 & 1.8424e-13 & 2.0132e-10 \\ -1.5147e-13 & 1.3108e-14 & -8.2383e-14 & -1.5944e-14 & -1.7422e-11 \\ 9.5198e-13 & -8.2383e-14 & 5.1777e-13 & 1.0020e-13 & 1.0949e-10 \\ 1.8424e-13 & -1.5944e-14 & 1.0020e-13 & 1.9393e-14 & 2.1191e-11 \\ 2.0132e-10 & -1.7422e-11 & 1.0949e-10 & 2.1191e-11 & 2.3156e-08 \end{bmatrix}
 \end{aligned} \tag{176}$$

Finally we present the numerical results for one of the polynomial blends which contains 1,550 measurements given by Equation 177.

$$\begin{aligned}
 & \begin{bmatrix} \sigma_{AA}^2 & \sigma_{AB}^2 & \sigma_{AC}^2 & \sigma_{AD}^2 \\ \sigma_{BA}^2 & \sigma_{BB}^2 & \sigma_{BC}^2 & \sigma_{BD}^2 \\ \sigma_{CA}^2 & \sigma_{CB}^2 & \sigma_{CC}^2 & \sigma_{CD}^2 \\ \sigma_{DA}^2 & \sigma_{DB}^2 & \sigma_{DC}^2 & \sigma_{DD}^2 \end{bmatrix} \\
 &= \begin{bmatrix} 1.3166e-16 & 1.5882e-14 & -1.9417e-12 & -2.3266e-10 \\ 1.5882e-14 & 1.9159e-12 & -2.3423e-10 & -2.8065e-08 \\ -1.9417e-12 & -2.3423e-10 & 2.8636e-08 & 3.4311e-06 \\ -2.3266e-10 & -2.8065e-08 & 3.4311e-06 & 0.0004 \end{bmatrix} \quad (177)
 \end{aligned}$$

### 5.3 Data Set #2 Results

In this section we process the second data set which contains 1,625 tracks with a total of 88,685 measurements. The second data set as shown in Figure 52 was expected to be more complex in all aspects of the processing. Not only were more lines extracted but additional attempts at extracting were taken by varying the parameters in the Hough section of the GUI. The extraction process took a total of seven steps outlined as follows:

- 1) The initial processing, wherein no intervention was taken, the first 400 tracks were processed with a step of 10 and then the remaining tracks were processed as a whole.
- 2) The first cleaning stage, removal of several areas, merges of a few lines and polynomial blends.
- 3) The second extraction stage, altered the parameters to attempt to extract smaller, finer line segments.
- 4) Second cleaning stage, additional blends created.
- 5) Third extraction stage, additional finer, smaller segments identified.
- 6) Final cleaning stage.
- 7) One last association to associate necessary data.

In the next subsection we show the graphical results in the step by step manner and then present an example of the Cramer Rao lower bounds estimate for one set of extracted features' parameters (i.e. a line segment, the ellipse, and a polynomial).

### 5.3.1 Graphical Results

The second data set was completely processed without manual intervention (i.e. merging or ellipse finding). This data set as shown in Figure 52 contains several areas of interest which may create issues (tightly spaced lines, sharp curves). Primarily we expect there to be a significant increase in the number of extracted segments. In addition to this increase the completeness of the extracted network may not be as accurate as data set one. This is due to the lack of measurements in some areas, which causes the Hough's failure to extract segments in the area. The initial extraction is shown in Figure 60.

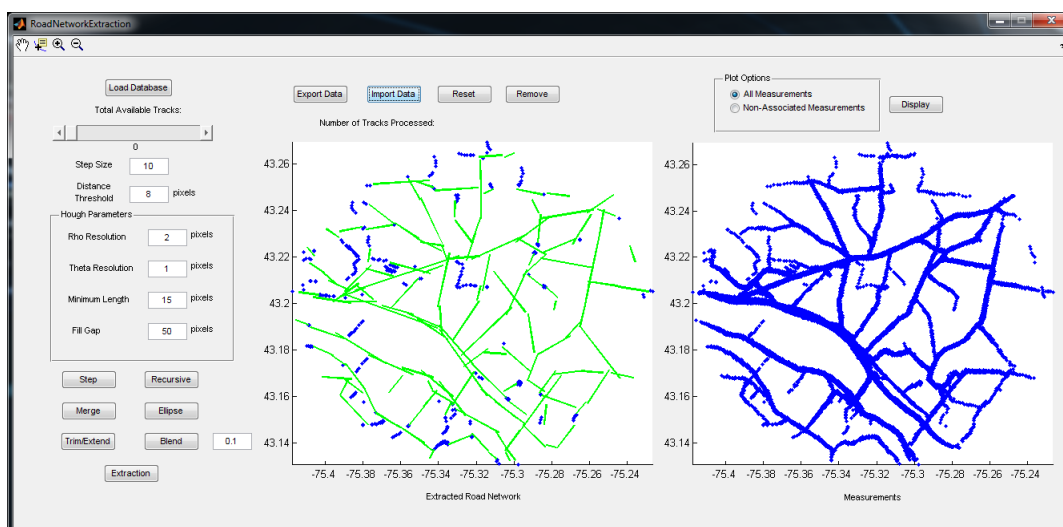


Fig. 60: Data Set #2 Phase One

The line extraction portion of the algorithm took approximately 25 minutes to complete. As with the first data set the extracted line segment data structure was stored before any additional functions were implemented. Prior to any removal or merging of line segments, the data structure contains 141 individual line segments. The next step was the beginning of the manual intervention to clean up the road network. This stage required the removal of several line segments so that we could attempt to extract a better fit for some of the curved regions as well as merging and blending. Figure 61 shows the results of the first stage of manual interactions taken.

The newly modified data structure now contains 105 line segments and 57 polynomial blends. With the second phase completed the extraction was attempted again by reducing the quantization

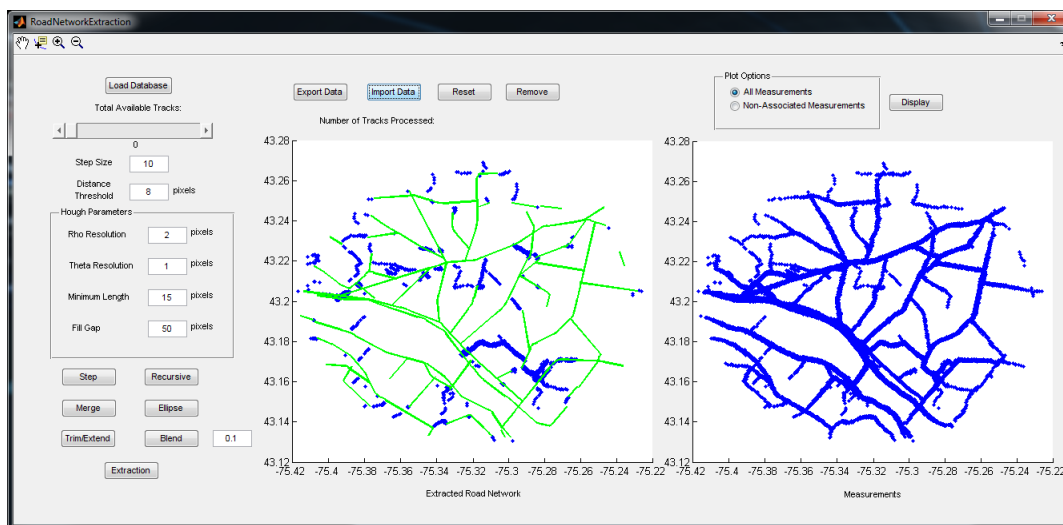


Fig. 61: Data Set #2 Phase Two

of  $\theta$  from 1 to 0.25 and the  $\rho$  from 2 to 1. In addition the value of the minimum length parameter was reduced to 10 to identify smaller lines and the fill gap threshold was reduced to 30. The result of this extraction is shown in Figure 62 where the discrepancies can be seen by comparing against Figure 61, the areas where the lines were clearly removed with a large set of measurements in the left plot.

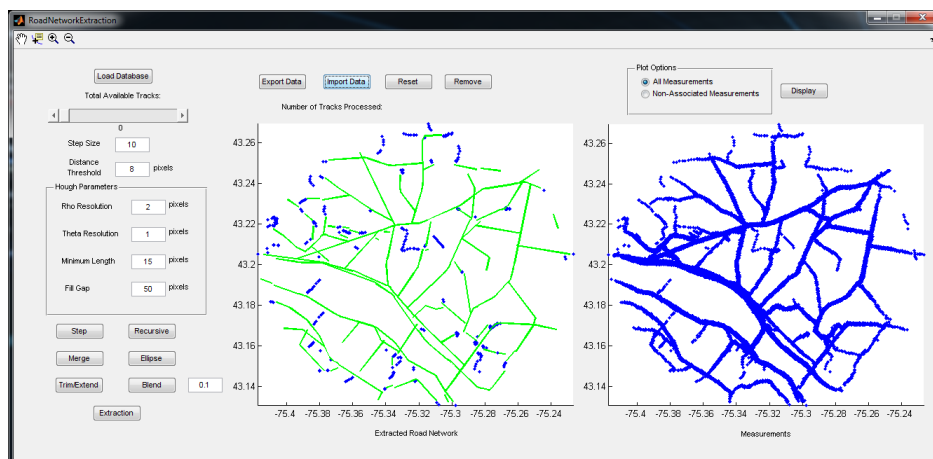


Fig. 62: Data Set #2 Phase Three

An additional 24 line segments have been identified after the extraction process was completed. Again, the user implemented a few merges as well as polynomial blends in an attempt to clean up and more accurately define the road network. This is shown in Figure 63.

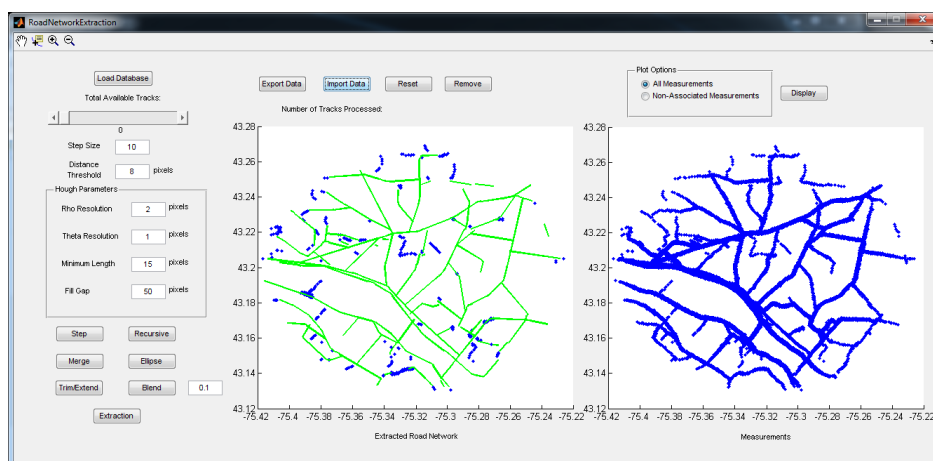


Fig. 63: Data Set #2 Phase Four

This newly modified data structure contains 126 line segments and 64 polynomial blends. In the next and final extraction phase, only the minimum length threshold was changed from 10 to 5 allowing for very small line segments to be identified. This resulted in the identification of an additional 17 line segments which are much more difficult to identify in the plot due to the small size. Figure 64 shows the results of the final extraction process.

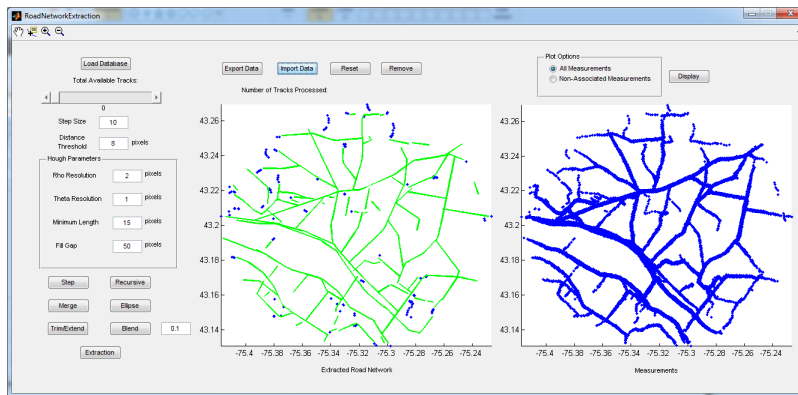


Fig. 64: Data Set #2 Phase Five

Additional merges and blends were implemented and this new structure contains 139 line segments and 74 polynomials. Figure 65 shows the results of the final manual actions taken.

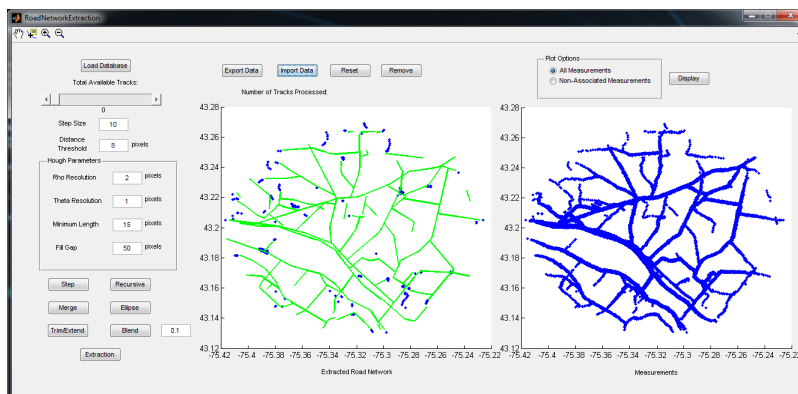


Fig. 65: Data Set #2 Phase Six



A final measure was taken due to a few last changes in the data structure and to attempt to associate remaining data. The distance threshold was increased from 8 pixels to 10 and the final results of the manual interactions and extraction process are shown in Figure 66.

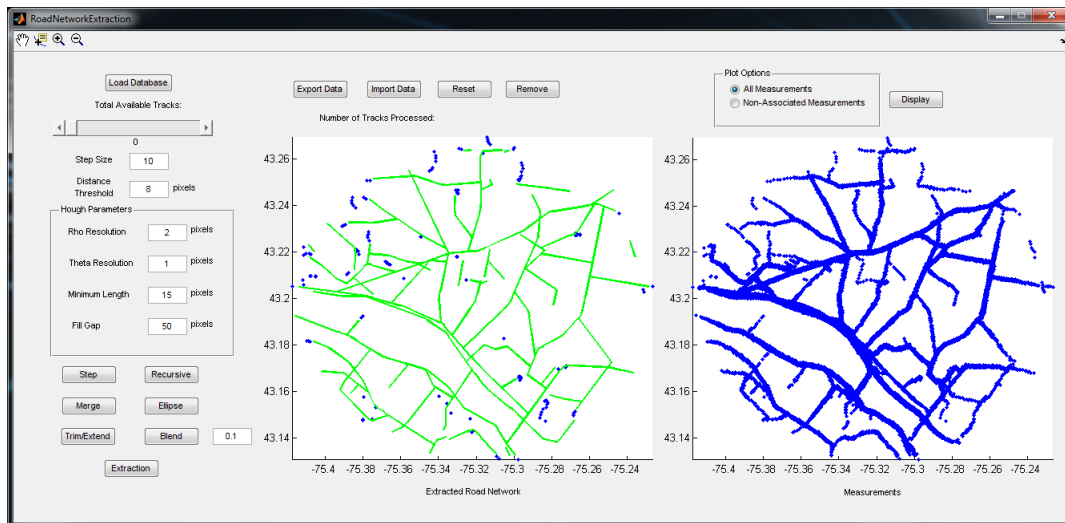


Fig. 66: Data Set #2 Final Extraction Results

This final data structure contains 139 line segments and 74 polynomials, the same as in the previous phase since no additional extraction was done. From stage one, the preliminary processing of the data, until stage seven approximately 2.5 hours passed. Therefore, to process the entire data set two from start to finish took approximately 2 hours and 45 minutes. A total of 177 measurements remained after all phases of processing resulting a 99.8% of the data being associated with an extracted feature (88,508/88,685).

### 5.3.2 Uncertainty Analysis

As with the first data set we present here the uncertainty in the estimated parameters using the various equations from Section 3.11. First we present the results for the estimated CRLB matrix corresponding to the line estimate's parameters. Figure 67 shows the three sigma bounded region for the lines, which again due to the number of measurements associated with each of the lines, converges to very finite values for most lines. A few of the lines have very few associated measurements and the bounds are very lax.

We supplement Figure 67 with an example of a line's CRLB estimate. Equation 178 refers to

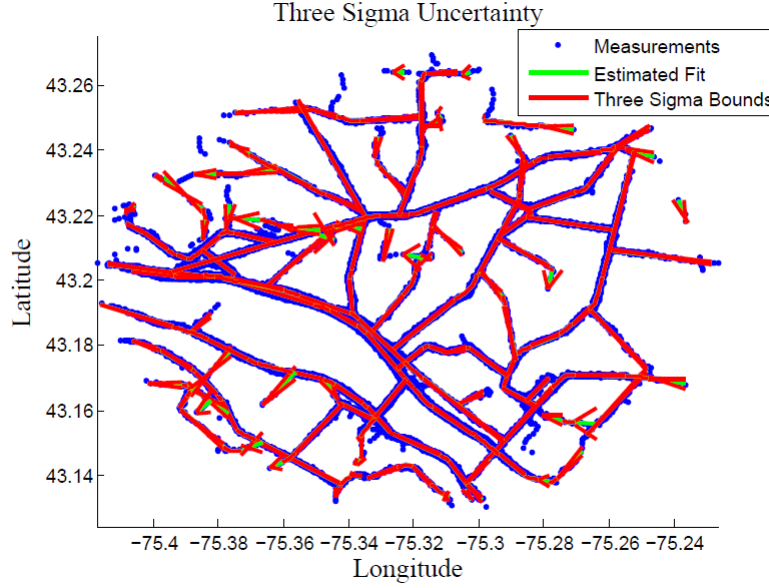


Fig. 67: Data Set #2 Line CRLB

a line which has 1,034 measurements associated with it. Since there are less measurements in this data set as well as more lines when compared against the first data set, the number of measurements associated with a line segment has significantly decreased to around 1,000 at most.

$$\begin{bmatrix} \sigma_{mm}^2 & \sigma_{mb}^2 & \sigma_{mx_t}^2 \\ \sigma_{bm}^2 & \sigma_{bb}^2 & \sigma_{bx_t}^2 \\ \sigma_{x_tm}^2 & \sigma_{x_tb}^2 & \sigma_{x_tx_t}^2 \end{bmatrix} = \begin{bmatrix} 2.2541e-05 & 3.5129e-09 & -1.7855e-09 \\ 3.5129e-09 & 3.6176e-10 & -1.2573e-10 \\ -1.7855e-09 & -1.2573e-10 & 1.4242e-10 \end{bmatrix} \quad (178)$$

We must then use the Jacobian transformation to obtain the covariance in terms of  $x$  and  $y$ . A single measurement's covariance matrix is given by 179:

$$\begin{bmatrix} \sigma_{xx}^2 & \sigma_{xy}^2 \\ \sigma_{yx}^2 & \sigma_{yy}^2 \end{bmatrix} = 1.0e-06 \times \begin{bmatrix} 0.3850 & -0.4209 \\ -0.4209 & 0.8059 \end{bmatrix} \quad (179)$$

The Jacobian transformed CRLB is then added to the measurement covariance defined in Equation 179 to produce the final covariance in the measurement including the covariance in the estimate given by Equation 180.

$$\begin{bmatrix} \sigma_{xx}^2 & \sigma_{xy}^2 \\ \sigma_{yx}^2 & \sigma_{yy}^2 \end{bmatrix} = 1.0e-06 \times \begin{bmatrix} 0.3852 & -0.4209 \\ -0.4209 & 0.8089 \end{bmatrix} \quad (180)$$

Figure 68 shows a few measurement's one sigma ellipses with the line estimate.

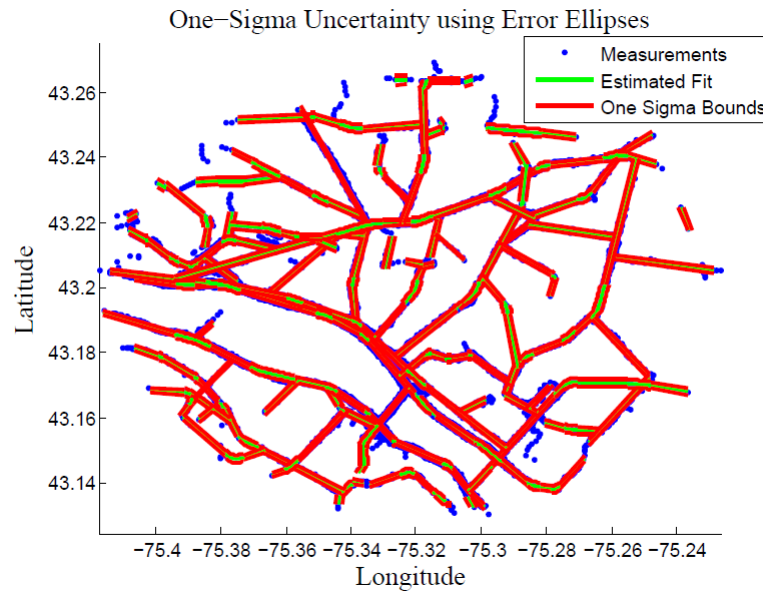


Fig. 68: Data Set #2 Error Ellipses One Sigma

Using the error ellipses we once again compute the mean covariance in  $x$  and  $y$ . This covariance is then used to represent a mean error in the line using one sigma we can see from Figure 69 that this encompasses the majority of the data associated with each of the line estimates.

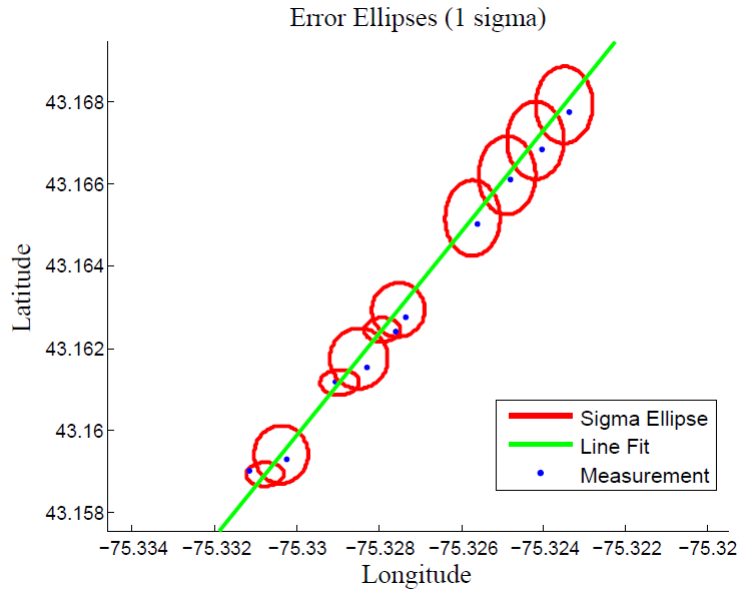


Fig. 69: Data Set #2 Mean Line Uncertainty

There are no ellipses identified in this data set but there is a significant number of polynomial blends. A polynomial blend in this data set typical contains from 400 to 1,000 measurements. We present the results for a polynomial which contains 1,032 associated measurements in Equation 181:

$$\begin{aligned}
 & \begin{bmatrix} \sigma_{AA}^2 & \sigma_{AB}^2 & \sigma_{AC}^2 & \sigma_{AD}^2 \\ \sigma_{BA}^2 & \sigma_{BB}^2 & \sigma_{BC}^2 & \sigma_{BD}^2 \\ \sigma_{CA}^2 & \sigma_{CB}^2 & \sigma_{CC}^2 & \sigma_{CD}^2 \\ \sigma_{DA}^2 & \sigma_{DB}^2 & \sigma_{DC}^2 & \sigma_{DD}^2 \end{bmatrix} \\
 &= \begin{bmatrix} 3.9162e-12 & 2.9909e-10 & -2.0009e-08 & -1.5304e-06 \\ 2.9909e-10 & 2.2842e-08 & -1.5281e-06 & -0.0001 \\ -2.0009e-08 & -1.5281e-06 & 0.0001 & 0.0078 \\ -1.5304e-06 & -0.0001 & 0.0078 & 0.5980 \end{bmatrix} \quad (181)
 \end{aligned}$$

## 6.0 CONCLUSIONS

Many concepts have been brought together throughout the production of this report. The Hough transform was thoroughly explained and derived in Section 4.3.4. In addition to the standard line Hough transform the circle and ellipse transforms were explained and the circle case was implemented. The Least Squares, Total Least Squares, and Least Squares ellipse algorithms were defined in Section 4.4.2. The purposes of each was explained and the equations were derived with the appropriate assumptions. Section 3.11 derived the uncertainty in the estimates developed for the line, third order polynomial, and the ellipse. These derivations were accompanied with examples and Monte Carlo simulations which showed that the solutions converged.

Sections 4.3.4-3.11 developed concepts and equations which were required during the production of the Graphical User Interface defined in Section 4.0. The GUI allows for user interaction at different stages. The accuracy of the extracted road network not only depends on the automatic extraction of the line segments via the Hough transform but also on the time spent by the user. As the results showed, 99.98% of the data in the first data set was associated with geometric features and in the second data set the association was 99.8%.

Full automation of an accurate road network would be extremely complex. The ability to identify an ellipse in an image automatically was briefly explored in Section 4.3.4, however the approaches available would require a significant amount of computation time as they compare every pixel's gradient direction or each line's orientation. In addition to the ellipse extraction, the merging, blending, trimming, and extending processes would need to be automated. The issues with each of these is that thresholds would need to be established corresponding to the line proximity and orientations.

The desired output of this work was a user interface which could be easily utilized and depending on the time available would provide an accurate road network (accuracy directly proportional to time spent). The results obtained as far as associated measurements was very good however these results are directly correlated with the amount of time spent on the processing.

The Euclidean distance threshold and the endpoint tolerances work well for the two data sets currently available. These values may not work well with additional data sets and can easily be incorporated as an additional user input into the GUI initialization process.

The algorithm developed currently has no potential ability to be computed in parallel. This is due to the recursive fashion in which a single line is identified and the points are removed from the image and then another line is identified with the remaining data. However there are areas in which the algorithm can be cleaned up and better coded. In addition to cleaning up the code, MatLab is not an efficient image processing package. As seen with the Hough transform, the MatLab Hough function is actually a compiled mexfile (C code).

Full automation as stated previously would be extremely complex, however the merging process may be automated by identifying lines with a similar orientation (slope and intercept) in addition to the use of thresholds.

Another possible area of improvement would be with the blending function. Currently a third order polynomial is used which considers the last xx%, defined by the user, of each of the two lines selected. Other options exists such as higher or lower order polynomials. The third order was chosen as it appeared best for the two data sets.

During the association phase of the extraction process, the distance from the current measurement to each of the features is computed and the minimal distance is then found. This area of the algorithm appears to be sluggish in its implementation and further adaptation is required to improve its efficiency.

## REFERENCES

- [1] Tupin, F., Maitre, H., Mangin, J., Nicolas, J. and Pechersky, E., "Detection of Linear Features in SAR Images: Application to Road Network Extraction." *IEEE Transactions on Geoscience and Remote Sensing*, vol. 36, 1998.
- [2] Koch, W., Koller, J. and Ulmke, M., "Ground Target Tracking and Road Map Extraction." *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 61, pg. 197-208, 2006.
- [3] Blackman, S., "Multiple Hypothesis Tracking for Multiple Target Tracking." *IEEE Aerospace and Electronic Systems Magazine*, vol. 19, pg. 5-18, Jan. 2004.
- [4] Baumgartner, A., Hinz, S. and Wiedemann, C., "Efficient methods and interfaces for road tracking," *Int. Arch. Photogramm. Remote Sens.*, vol. 34, pg. 28-31, 2002.
- [5] Zhou, Y.T., Venkateswar, V. and Chellapa, R., "Edge detection and linear feature extraction using a 2-D random field model," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 11, pg. 84-95, Jan. 1989.
- [6] Amo, M., Martinez, F. and Torre, M., "Road extraction from aerial images using a region competition algorithm," *IEEE Trans. Image Process.*, vol. 15, pg. 1192-1201, May 2006.
- [7] Gamba, P., Dell'Acqua, F. and Lisini, G., "Improving urban road extraction in high-resolution images exploiting directional filtering, perceptual grouping, and simple topological concepts," *IEEE Geosci. Remote Sens. Lett.*, vol. 3, pg. 387-391, Jul. 2006.
- [8] Amberg, V., Coulon, M., Marthon, P. and Spigai, M., "Structure extraction from high resolution SAR data on urban areas," in *Proc. IEEE Geosci. Remote Sens. Symp.*, vol. 3, pg. 1784-1787, 2004.
- [9] Hu, J., Razdan, A., Femiani, J., Cui, M., and Wonka, P. "Road Network Extraction and Intersection Detection from Aerial Images by Tracking Road Footprints," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 45, pg. 4144-4157, Dec. 2007.
- [10] Sklarz, S.E., Novoselsky, A., and Dorfman, M., "Incremental Fusion of GMTI Tracks for Road Map Estimation," *2008 11th International Conference on Information Fusion*, pp. 1169-1175. 2008.
- [11] Shackelford A., and Davis, C., "Urban Road Network Extraction from High-Resolution Multispectral Data," *2nd GRSS/ISPRS Joint Workshop on Remote Sensing and Data Fusion over Urban Areas*, pg. 142-146, 2003.
- [12] Duda, R. and Hart, P., "Use of the Hough Transformation to Detect Lines and Curves in Pictures," *Comm. ACM*, vol. 15, pg. 11-15, Jan. 1972.
- [13] Guil, N., and Zapata, E., "Lower Order Circle and Ellipse Hough Transform," *Pattern Recognition*, vol. 30, pg. 1729-1744. 1997.
- [14] Tsuji, Saburo, and Matsumoto, "Detection of Ellipses by a Modified Hough Transformation," *IEEE Transactions on Computers*, vol. 27, pg. 777-781, 1978.
- [15] Aguado, A., and Nixon, M., "A New Hough Transform Mapping for Ellipse Detection," *Technical Report*, 1995.
- [16] Crassidis, J., and Cheng, Y., "Error-Covariance Analysis of the Total Least Square Problem," *Proceedings of the AIAA Guidance, Navigation and Control Conference*, pg. 8-11, 2011.
- [17] Fitzgibbon, A. and Fischer, R., "A buyer's guide to conic fitting," *Proc. of the British Machine Vision Conference*, pg. 265-271, 1995.
- [18] Halir, R. and Flusser, J., "Numerically stable direct least squares fitting of ellipses," *The Sixth International Conference in Central Europe*, 1998.

- [19] Weisstein, E., "Ellipse," *MathWorld A Wolfram Web Resource*, <http://mathworld.wolfram.com/Ellipse.html>
- [20] Weisstein, E., "Point Line Distance 2-Dimensional," *MathWorld A Wolfram Web Resource*, <http://mathworld.wolfram.com/Point-LineDistance2-Dimensional.html>
- [21] Chernov, N. and Lesort, C., "Statistical Efficiency of Curve Fitting Algorithm," *Computational Statistics and Data Analysis*, vol. 47, pg. 713-728, Nov. 2004.
- [22] Al-Sharadqah, A. and Chernov, N. "A Doubly Optimal Ellipse Fit," *Computational Statistics and Data Analysis*, vol. 56, pg. 2771-2781, Sept. 2012.
- [23] Ravindran, A., Ragsdell, K., and Reklaitis, G., "Engineering Optimization Methods and Applications," Hoboken, New Jersey. Wiley and Sons. 2006. Print.
- [24] [http://www.brookscole.com/math\\_d/templates/student\\_resources/AdditionalTopics/RotationofAxes.pdf](http://www.brookscole.com/math_d/templates/student_resources/AdditionalTopics/RotationofAxes.pdf)
- [25] Long, D. "ellipse.m," MATLAB Central File Exchange. Oct. 9, 1998.
- [26] Davis, T. "getversion: find MATLAB version number as a double," MATLAB Central File Exchange. Nov. 1, 2007.



## **List of Acronyms**

CHT Circle Hough Transform

EHT Ellipse Hough Transform

FaHT Fast Hough Transform

FHT Fuzzy Hough Transform

GMTI Ground Moving Target Indicator

GPS Global Positioning System

GUI Graphical User Interface

LS Least Squares

MHT Multiple Hypothesis Tracking

RMSE Root Mean Square Error

SAR Synthetic Aperture Radar

TLS Total Least Squares